实用第一 智慧密集 Practicability First, Intelligence Intensive

# 电脑编程技巧与维护

COMPUTER PROGRAMMING SKILLS & MAINTENANCE

http://www.comprg.com.cn

上 6月 2013年6月03日

每期定价:11.00元 全年定价:264.00元
《电脑编程技巧与维护》 杂志社出版
刊号: ISSN 1006-4052 / CN11-3411/TP
广告许可证 京海工商广字0151

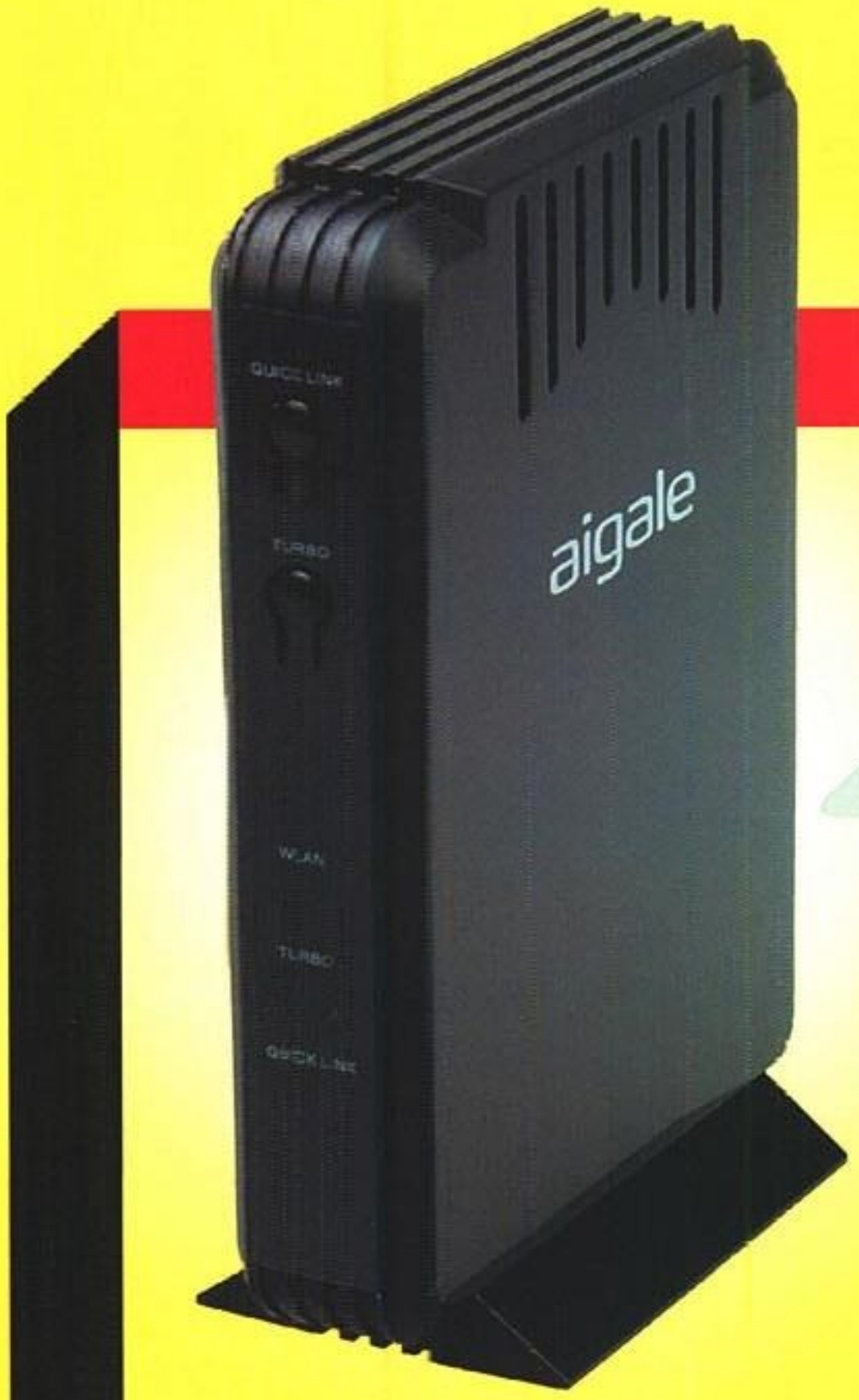
国家级科技期刊 中国学术期刊综合评价数据库统计源期刊 中国核心期刊（遴选）数据库收录期刊



第95页.《基于DirectUI架构的中海油软件平台的用户体验设计与实现》

# 抢先Hold住PCWorld

## 即可精巧“联”通科技未来！




**现在邮购2013年**

《微电脑世界》**全年杂志**

**即得一个价值149元**

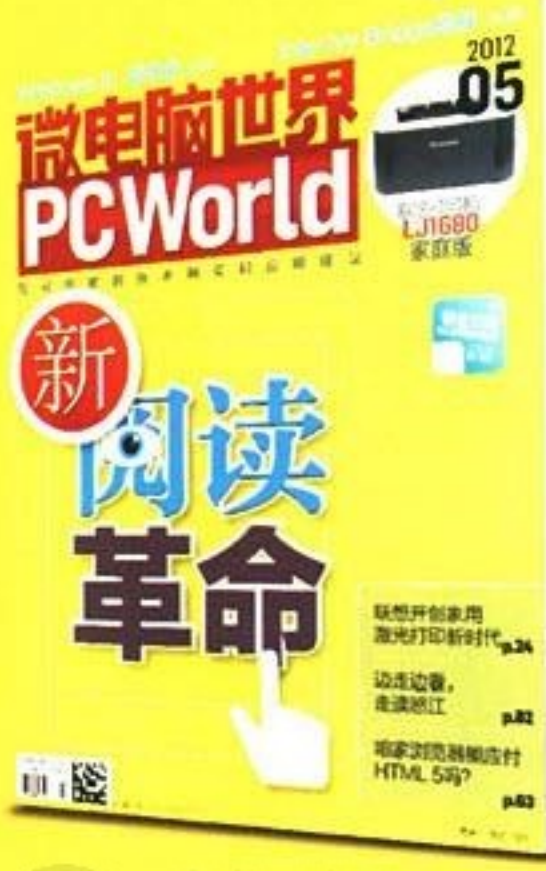
**海联达Ai-R100 极风**

# 无线路由器

轻松联通您的智能终端，
让您尊享全球IT资深顾问
随时随地的贴身资讯服务。

飞信 Fetion

**用户可登陆：**
http://sms.feixin.10086.cn/Subscribe/getInfo/id/215 订阅
中国移动用户发送ZZ到125200915，即可订阅




**汇款地址：**北京市123信箱，收款人：微电脑世界杂志，邮编：100036
**杂志定价：**144元/年（12元/月）
**活动咨询：**周一到周五，9：00~11：30，13：00~17：30
**电话：**010-88230131
**杂志社现场订阅地址：**北京市海淀区万寿路翠微中里14号楼
**在线订阅：**http://www.pcworld.com.cn/about/ebuy/pay.html

**活动说明：**

**活动时间：**2012年8月10日~2013年6月30日（邮局汇款以邮戳为准）
在汇款单附言栏注明“2013年微电脑世界”，同时留下联系电话。
如需发票，请在汇款单附言注明“发票”以及发票抬头，过后将不能补开。
本活动仅限于在杂志社订阅的读者，邮局订阅等其他渠道不参加此活动。
由于本次活动涉及奖品发放，参与活动的读者将不能中途退订。
**邮费：**平寄邮费由杂志社承担，如需挂号，每本另加3元挂号费，汇款时一并汇上，并注明挂号字样。

2013 年第 11 期
6 月(上)

# 电脑编程技巧与维护

总第 281 期　　1994 年 7 月创刊　　(半月刊)


地　　址：北京市海淀区长春桥路 5 号
6 号楼 1209 室
投 稿 信 箱：gaojian@comprg.com.cn
gaojian@comprg.sina.net
编辑部信箱：gaojian@comprg.com.cn
发行部信箱：zzsfx@vip.sina.com
网　　址：http://www.comprg.com.cn
邮　　编：100089
电　　话：(010) 82561037
传　　真：(010) 82561614
照　　排：《电脑编程技巧与维护》
杂志社电脑排版部
印 刷 厂：北京慧美印刷有限公司
订 阅 处：全国各地发行局
国内总发行：北京报刊发行局
邮 发 代 号：82-715
国外发行代号：M6232

刊　　号：ISSN 1006-4052
CN11-3411/TP

广告订可证：京海工商广字 0151 号
全 年 定 价：264 元
每 期 定 价：11 元

# 电脑编程技巧与维护(半月刊)

每月3日出版


实用第一 智慧密集

# 目次

## 基于百度云首款穿戴式设备咕咚手环即将亮相

国内一款穿戴式设备—咕咚手环不久将亮相。主打的三大功能：运动状况提醒、睡眠监测、智能无声唤醒。同时，该手环是首款基于百度云开发的便携式设备。

据了解，咕咚手环能支持运动提醒，还可通过记录睡眠，在最理想的时刻将佩戴者唤醒。用户可将该手环穿戴在手腕上，24 小时监测每天活动量及睡眠情况。此款手环还与百度云结合，用户可以把运动手环中所记录的数据实时汇总到百度云端，随时记录察看。

作为咕咚手环的开发方，咕咚网并无大规模的硬件基础，而是通过借助百度云的大数据量处理、高并发支撑、及安全存储、可伸缩等特性，对这一智能穿戴式设备进行研发。

据透露，咕咚手环 6 月初起将在咕咚网官方商城和天猫旗舰店发售。用户可以在咕咚网上进行数据承载、展示，并可在社交网站上进行分享。当处于运动模式时，该手环能 24 小时记录佩戴者的活动情况，以里程、步数和卡路里为单位，令佩戴者明晰一整天内，运动了多少距离，消耗了多少卡路里，为热衷减肥和运动的用户提供了实时监测服务。

切换至睡眠模式时，除了能监测睡眠质量，手环还将根据使用者睡眠深浅状态，在应该叫醒的时间段中的浅睡状态下通过震动来唤醒佩戴者。

知情人士透露，未来咕咚网与百度云合作还将深入，蓝牙体重秤、蓝牙自行车码表等一系列手机智能配件将会陆续推出，而这些配件都需要移动 APP 的支持以及云存储服务支持。业界人士分析，智能穿戴式设备是传统硬件、新交互技术（语音、手势识别、眼球识别、骨传导等技术）与云应用服务的结合体，基础架构、产品能力往往会迁移到云端，利用云服务、多屏云端同步来实现。

## NFC 在安防领域的应用：数码钥匙

近距离无线通信 (NFC) 是一项适用于门禁系统的技术，这种近距离无线通信标准能够在几厘米的距离内实现设备间的数据交换。NFC 还完全符合管理非接触式智能卡的 ISO 标准，这是其成为理想平台的一大显著特点。通过使用配备 NFC 技术的手机携带便携式身份凭证卡，然后以无线方式由读卡器读取，用户只需在读卡器前出示手机即可开门。据研究机构 IHSiSuppli 预测，2015 年，制造商将出厂约 5.5 亿部支持 NFC 的手机。

NFC 虚拟凭证卡的最简单模式就是复制现行卡片内的门禁原则。手机将身份信息传递给读卡器，后者又传送给现有的门禁系统，最后打开门。这样，无需使用钥匙或智能卡，就可提供更安全、更便携的方式来配置、监控和修改凭证卡安全参数，不仅消除了凭证卡被复制的风险，而且还可在必要时临时分发凭证卡，若丢失或被盗也可取消凭证卡。

新一代智能卡技术的发展，使得企业能够通过将门禁和电脑桌面登录整合到单一的身份识别平台，来保护设施、人员和资产。这种集成多应用的门禁解决方案不仅可用于传统的证卡和读卡器，还将用于移动设备 (包括手机)。这些手机使用无线近距离通信 (NFC) 技术来接收及出示以往寄存在非接触式智能卡的虚拟凭证卡，实现开门、电子支付和安全读取数据等应用。

NFC 移动访问设备具有智能特性，能够验证个人身份信息和其他相关访问规则，从而降低了未来对门禁读卡器 (及门锁) 的智能性和连接功能的要求。此外，NFC 手机将通过使用加密的安全通信发送认证信息至读卡器，实现门禁的控出准入。这个过程中，读卡器只是需要解读用于开门的加密命令。—读卡器或锁具未来可免连接到控制面板或服务器，显著降低读卡器或锁具的部署成本。

融合门禁和电脑桌面登录的集成多应用解决方案是门禁行业的大势所趋，并正在创造大量新的市场机会。随着此类解决方案不断迁移到 NFC 智能手机等移动平台，它们有机会成为客户的首选方案。

## 可随手生成的触摸界面

据物理学家组织网近日报道，最近，美国卡内基·梅隆大学（CMU）人机互动研究所（HCII）研究人员开发出一种深度摄像/投影系统，让人们能通过手势在日常生活所接触的任何表面上生成计算机界面。研究人员在 CHI 2013（计算机系统中的人类因素）大会上汇报了这一成果。

此前有研究人员曾展示过一种深度摄像系统，能与投影仪结合，将任何表面转移到一个触摸屏上，而卡内基·梅隆大学最新演示是用户能按照自己意愿，通过手的动作来随意生成触摸界面。

最新成果被称为世界工具系统（WorldKit），让人们能摩擦沙发来遥控电视，或在办公室门上挥手设置日历，而且别人还能在这日历上继续进一步操作。这些特殊界面还可以通过肢体动作加以移动、修改或删除，使它们变得高度个人化。

研究人员用装在天花板上的摄像机和投影仪来记录室内各种几何形状信息，感知手势并能随意在任何表面投下画面。HCII 博士生罗伯特·肖指出，今后 WorldKit 并不需要这么复杂的装置。“深度传感器正变得越来越好，投影仪也在变得越来越小。按照我们的展望，用一种互动‘灯泡’——就像普通电灯组件那样组装在一起的小型设备，能固定或移动到任何界面上。”

该系统不需要事先校准，它能自动调节感知系统，把画面投射到所选择表面的方向。用户可以整合开关、信息栏、指示灯和多种其他菜单的接口设计。据 WorldKit 小组预测，其最终可能成为一种能按用户需要专门定制的手势接口设计。

# 利用语义网技术实现铁路交通的地理语义查询（一）

## ——C# 中地理数据的网络获取和本体构建

董 志

**摘 要**：介绍了利用网络地理信息服务获取到有关城市特定区域内有关火车站的 POI 点信息，并根据这些信息，在 C# 环境下，基于 Java 虚拟机，使用 Jena API 设计了本体的类、属性、个体的操作类，为铁路交通中火车站点、线路及所在城市的本体构建及本体推理做准备。

**关键词**：语义网；Web3.0 技术；Baidu MapAPI 编程；地理数据获取；SQLite 引擎；Jena 工具；地理本体；本体构建，IKVM.NET 虚拟机

经过近 10 多年的研究与发展，作为 Web3.0 重要组成部分的语义网已经走出实验室进入工程实践阶段。由于万维网上已产生了浩瀚的网络信息和知识资源，寻找人们所需要的准确信息常常耗费大量人力精力。提供网络信息的语义半自动化或自动化处理已迫在眉睫。这就使语义网是成为 Web 3.0 最有希望的基础技术。它的核心思想是：通过给万维网上的文档 (如: HTML) 添加能够被计算机所理解的语义 (Meta data)，从而使整个互联网成为一个通用的信息交换媒介。语义网与本体技术实际上是人类知识领域的概念标准化运动，这就涉及到逻辑描述（Description Logics）和推理（reasoning）技术。由于描述网络数据的需要，科学家们开发了一系列元数据描述语言，如 RDF/RDFS 等。出于建模后进行语义分析与推理的需要，科学家规定了本体描述语言 （如 OWL），并开发了种种特定领域的本体（Ontology）。所谓本体，可以简单地将它理解成特定知识领域中满足共同约定的常识部分，这对于特定领域信息分类是必要的一步。

接下来要实现的功能就是主要利用语义网技术，在 C# 环境中基于 IKVM 的 Jena 接口来实现铁路交通的语义查询功能，该功能结合了地理信息系统的周边领域，获取城市中心范围内与火车站相关的 POI 信息，存储于 SQLite 数据库中，然后利用 Jena 推理进行“包含于”某城市的语义查询。本程序实验数据主要为在线调用百度地图数据。程序最后的实现效果如图 1 所示。

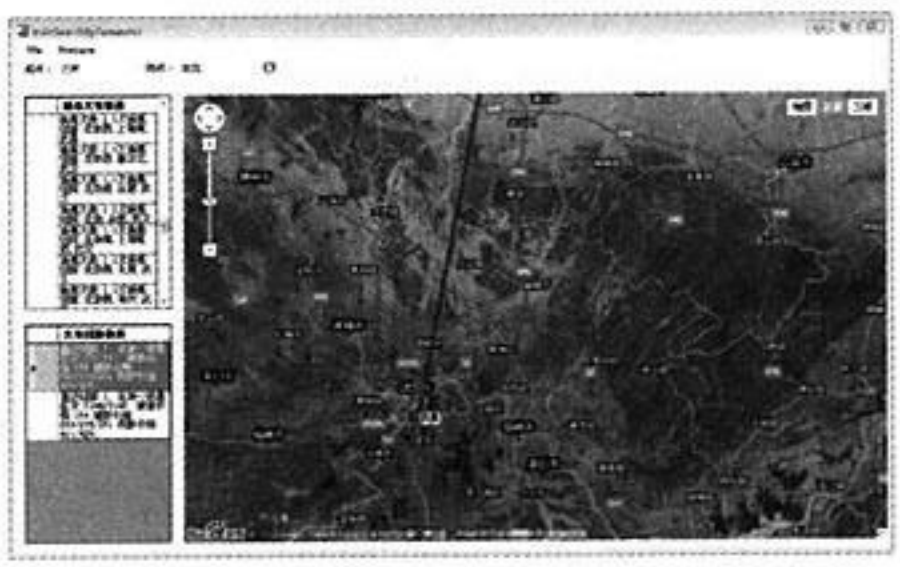


**图 1 铁路交通换乘语义查询并在地图上可视化效果**

本程序使用的是 SQLite 数据引擎和 Java API 编程，因此很容易移植到 Android 系统。

功能实现分为 3 篇介绍。本篇主要讨论了总体设计思路，介绍在 C# 中进行地理数据的网络获取、SQLite 数据操作类和本体操作类的设计，第二篇介绍从关系数据库中创建本体与定义推理规则。第三篇介绍集成本体推理结果并在地理空间中可视化。

## 1 问题的提出

传统的信息检索通常是通过关键词来实现的，其原理是：用户提出查询式—通常由若干个反映主题的词汇组成， 然后系统在数据库中将提问式与预存的文本关键词进行自动匹配， 两者相符的文本被检出。但是大量的事实证明， 这种通过词汇简单匹配检索出的结果并不是最优的。例如，在进行火车站点换乘查询时（目前大多数程序就是根据关键词查询），当用户查询“武汉”关键词，往往只能查询到包含“武汉”字符的记录，而使用者的真实意图可能还需要知道武汉区域的“武昌站”、“汉口站”。本程序就是要利用语义网技术解决这个问题。

## 2 设计思路

下文所述的具体实现方法是根据百度地图的 Web 服务进行城市中心位置的周边查询，将查询结果存储到 SQLite 数据中并与铁路交通数据相结合，利用运行于 IKVM.NET 上的 Jena API 创建铁路站点本体，将本体存储为 owl 文件；当用户进行火车站点查询时，基于创建完的本体模型进行规则推理和本体查询，得到语义查询结果并在网络电子地图上显示。方法设计思路如图 2 所示。

以下分别简单介绍一下百度地图 JavaScript API 地图引擎、SQLite 引擎、Jena 工具、IKVM.NET 虚拟机。

百度地图 JavaScript API 提供了地图基本功能（显示、平移、缩放、拖拽等）、兴趣点搜索、周边搜索、公交驾车路线搜索、逆/地理编码等，可以十分方便地获取地理信息数据并

搭建电子地图应用。

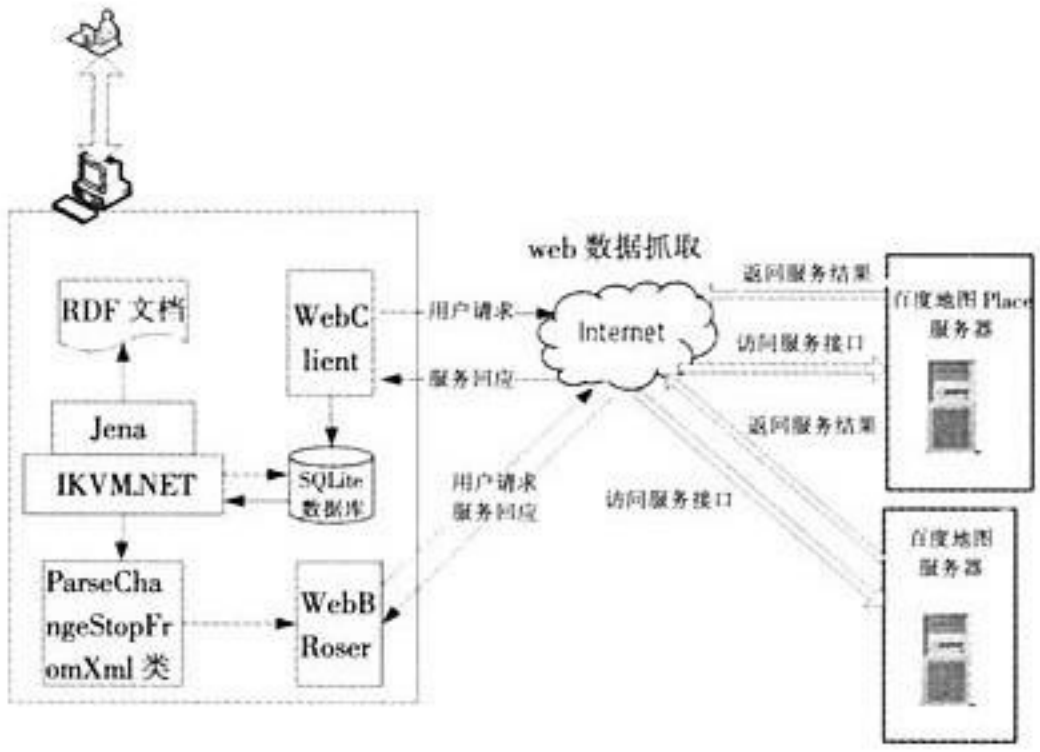



**图 2 语义网技术实现铁路交通查询的技术路线**

SQLite 是一款轻型的嵌入式数据库，它有着相当小的内存占用和高速的响应，遵守 ACID 的关联式数据库管理系统；它的设计目标是嵌入式的，而且目前已经在很多嵌入式产品中使用了它，在嵌入式设备中，可能只需要几百 KB 的内存就够了。它能够支持 Windows/Linux/Unix 等等主流的操作系统，同时能够跟很多程序语言相结合，比如 C#、PHP、Java 等，还有 ODBC 接口；比起 Mysql、PostgreSQL 这两款开源的数据库管理系统来讲，它的处理速度比他们都快。SQLite 3 版本已经发布，很多著名的公司（诸如 Adobe，Apple，Google，Sun，Symbian），开源项目（Mozilla，PHP，Python）都在产品中装配 SQLite。Android 中，SQLite 是被集成于 Android runtime，每个 Android 应用程序都可以使用 SQLite 数据库。

Jena 由 HP Labs（http://www.hpl.hp.com）开发的开源的 Java 开发工具包，用于 Semantic Web (语义网) 中的应用程序开发。Jena 框架主要包括：（1）以 RDF/XML、三元组形式读写 RDF：其内容包括 RDF 模型的创建、读写、查询等操作。（2）RDFS，OWL，DAML+OIL 等本体的操作：Jena 框架包含一个本体子系统（Ontology Subsystem），它提供的 API 允许处理基于 RDF 的本体数据。本体 API 与推理子系统结合可以从特定本体中提取信息。（3）利用数据库保存数据：Jena 2 允许将数据存储到硬盘中，或者是 OWL 文件，或者是在关系数据库中。这里处理的本体就是由 OWL 文件读入的。（4）查询模型 Jena 2 提供了 ARQ 查询引擎，它实现 SPARQL 查询语言和 RDQL，从而支持对模型的查询。（5）基于规则的推理：ena 2 支持基于规则的简单推理，其推理机制支持将推理器 (inference reasoners) 导入 Jena，创建模型时将推理器与模型关联以实现推理。

由于 Jena 是 Java 开发包，因此，在 C# 环境中使用 Jena，必须要使用虚拟机让 Java 程序和.NET 应用程序一起协同工作。程序选中的 C# 环境中 Java 虚拟机就是 IKVM.NET。

IKVM.NET 包含以下的部分：IKVM.Runtime.dll VM 运行时和所有支持代码。它包括以下的功能：Byte Code JIT 编译器和验证器，使用 JIT 将 Java Byte Code 编译为 CIL（C 中间语言）。对象模式映射结构，将.NET 中的 System.Object，System.String，System.Exception 映射为 Java 代码中的 java.lang.Object，java.lang.String，java.lang.Throwable。管理本地方法（在 Classpath 中）的.NET 重新实现。IKVM.GNU.Classpath.dll 被编译的 GNU Classpath 版本，它是由自由软件基金会实现的 Java 类库和一些 IKVM.NET 附加代码组成的。ikvmc.exe 静态编译器，被用来编译 java 类和 jar 使其成为.NET 汇编（静态模式）。ikvmstub.exe 一个从.NET 汇编生成存根类的工具，就如 javap 一样反编译.NET 汇编。

以上的 API 或者开源代码都是本程序要用到的。

## 3 使用 SQLite.NET 设计数据操作类

程序在.NET 使用的 wrapper 是 SQLite 3.7.14.，命名空间为 System.Data.SQLite，它只需要一个 dll，接口符合 ADO.Net 的定义，性能也不错，支持集成 VS2005、VS2008 和 VS2010，支持 .NET Framework 2.0–4.0，而且 Android 系统已经集成了 SQLite，这是个亮点。。

下面详细介绍怎么使用 SQLite.NET 实现一个操作 SQLite 数据库的类。因为 SQLite.NET 符合 ADO.NET 的规范，其使用方式基本和 OleDb、Odbc、SqlClient 等一致。但是，它的 SQL 语句略有区别，如“udpate”语句的字符串连结操作符为“||”，而不是有些数据库系统的“+”符号：

```
using System.Data;
using System.Data.SQLite;
//…
class SQLiteDB
  {
    private SQLiteConnection _conn;
    public SQLiteConnection conn
    {
      get { return _conn; }
      set { _conn = value; }
    }
    private string _connStr;
    public string connStr
{
  get { return _connStr; }
  set { _connStr = value; }
}
//处理连接字符串
public string GetDBConnStr(string DBCNString)
{
  if (DBCNString.Length <= 0 )
    DBCNString = "data/trainPOI";
  return DBCNString;
}
```

FOLLOW MASTER PROGRAM

```
//根据 SQL 语句得到 DataSet 对象
public DataSet getRSFromDb(string strQuery)
{
    try
    {
        if (conn.State == ConnectionState.Closed)
            openConnection("");
        DataSet rs = new DataSet();
                    SQLiteDataAdapter Adp = new
SQLiteDataAdapter(strQuery, _conn);
                Adp.Fill(rs);
                return rs;
            }
            catch (System.Exception e)
            {
                Console.Write(e.Message.ToString());
                return null;
            }
        }
        //根据 SQL 语句、连接字符串得到 DataSet 对象
          public static DataSet getRSFromDb (string
strQuery, string dbcn)
        {
            try
            {
      SQLiteConnection conn = new SQLiteConnection();
                conn.ConnectionString = dbcn;
                conn.Open();
                DataSet rs = new DataSet();
                            SQLiteDataAdapter Adp = new
SQLiteDataAdapter(strQuery, conn);
                Adp.Fill(rs);
                return rs;
            }
            catch (System.Exception e)
            {
                Console.Write(e.Message.ToString());
                return null;
            }
        }
        //得到 SQLite 数据表结构
    public DataTable getRSSchema(string collectionName)
    {
        try
        {
            if (conn.State == ConnectionState.Closed)
                openConnection("");
            DataTable dt = new DataTable();
            dt = conn.GetSchema(collectionName);
            return dt;
        }
        catch (System.Exception e)
        {
            Console.Write(e.Message.ToString());
            return null;
        }
    }
    //执行 SQLite 环境中的 SQL 语句
    public int ExecuteSQLiteSQL(string strSql)
    {
        try
        {
            if (conn.State == ConnectionState.Closed)
                openConnection("");
                    SQLiteCommand command = new
SQLiteCommand(strSql, _conn);
            int isOk = command.ExecuteNonQuery();
            return isOk;
        }
        catch (System.Exception e)
        {
            Console.Write(e.Message.ToString());
            return 0;
        }
    }
    //创建 SQLiteConnection 对象,并打开数据库
    public void openConnection(string connStr)
    {
        _conn = new SQLiteConnection();
        SQLiteConnectionStringBuilder sqLiteConnstr =
new SQLiteConnectionStringBuilder();
        _connStr = GetDBConnStr(connStr);
        sqLiteConnstr.DataSource = _connStr;
        sqLiteConnstr.Password = "";//设置密码
    _conn.ConnectionString = sqLiteConnstr.ToString();
        _conn.Open();
    }
    //关闭数据库
    public void closeConnection()
    {
        try
        {
            if (_conn.State == ConnectionState.Closed)
                _conn.Open();
        }
        catch(Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
    ~SQLiteDB()
    {
        closeConnection();
    }
```

}

没有空间定位信息的 SQLite 事例数据内容如图 3 所示。

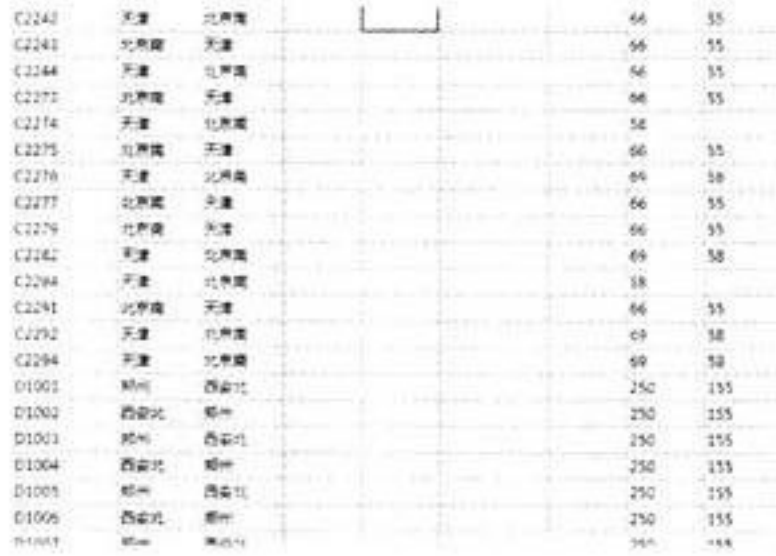


图 3 没有空间定位信息的 SQLite 火车站点数据

## 4 根据地图 Web 服务添加火车站点空间信息

程序中用到的测试数据是近期的部分铁道交通票价和主要城市名称，由于这两种数据只有属性信息，没有地理坐标信息，也没有空间拓扑关系中的“包含于”那个城市的关系信息，因此这些数据需要进行空间定位以获取这些信息。

程序利用.NET Framework 类库中的 WebClient 类调用 Baidu 地图的 Place Web 服务来完成空间信息获取。百度地图 Place 服务是一个供程序员调用的、http 形式的地图服务接口。它主要服务那些非网页程序的调用，例如 C# 、C++、Java 等开发语言都能发送 http 请求且能接收返回数据。用户只需在请求的 url 字串中拼接好关键词、检索区域和一些过滤条件，即可获取到想要的 POI 点的信息，包括该点的名称、地址、分类等信息。目前，Place API 提供的功能包括：矩形区域关键词检索、周边区域关键词检索和城市内关键字检索。

Place Web 服务地址及格式为：http://api.map.baidu.com/place/search?&query=关键词 &bounds=查询区域 &output=输出格式类型 &key=用户密钥。

程序用到的两个地理信息检索为：

（1）指定城市内检索（返回 xml 数据）

如：http://api.map.baidu.com/place/search?&query=北京 &region=北京 &output=xml&key=37492c0ee6f924cb5e934fa08c6b1676

返回事例数据为：

```
<?xml version="1.0" encoding="utf-8"?>
<PlaceSearchReaponse>
<status>OK</status>
<results>
<result>
<name>北京市</name>
<location>
<lat>lat 坐标</lat>
<lng>lng 坐标</lng>
</location>
<uid>7af0a1acef223dd0a1cc7f3</uid>
<result>
</results>
</PlaceSearchReapnse>
```

（2）周边区域检索（返回 xml 数据）

如：http://api.map.baidu.com/place/search?&query=火车站&location=39.914889,116.403874&radius=2000&output=xml&key=37492c0ee6f924cb5e934fa08c6b1676

返回事例数据如图 4 所示。

图 4 返回事例数据

程序的 WebClient 对象从 URI 标识的以上的地图服务资源接收脱密变形后数据，一边用城市内检索进行城市坐标定位，一边用周边区域检索对火车站点定位并确定“包含于”关系，程序具体实现如下。

```
using System.Data.SQLite;
using System.Net;
using System.Xml;
    class GetTrainLocationFromWeb
      {
        //创建数据库操作对象
        SQLiteDB sqliteDbObj = new SQLiteDB();
         //根据城市表 CapitalOfprovince 中的数据进行数据
//的空间定位并确定"包含于"的逻辑关系
        public void GetTrainLocationInfo()
        {
            sqliteDbObj.openConnection("");
             sqliteDbObj.ExecuteSQLiteSQL ("delete from
trainStopLocationCity");
                DataTable dt = sqliteDbObj.getRSFromDb
("select cityName from CapitalOfprovince").Tables[0];
              for (int i = 0; i < dt.Rows.Count; i++)//得到城市
//中心经纬度坐标
                getStopLocationInfo(dt.Rows[i][0].ToString());
              dt.Clear();
                    dt = sqliteDbObj.getRSFromDb("select
cityName,lat,lng from CapitalOfprovince").Tables[0];
              for (int i = 0; i < dt.Rows.Count; i++)//根据城市
//中心坐标获取城市与火车有关的带坐标兴趣点
                createStopLocationInfo(dt.Rows[i][0].ToString
(),dt.Rows[i][1].ToString(), dt.Rows[i][2].ToString());
```

```
        }
        private void getStopLocationInfo(string region)
        {
            string POISearchUrl = "http://api.map.baidu.
com/place/search&query = " + region + "&region = 北京
&output=xml&key=46b68150f22408ef4e4dcf3281ecb400";
//需要获取源代码的网页
            WebClient mapSebClient = new WebClient();
// 创建 WebClient 实例提供向 URI 标识的资源发送数据和从
//URI 标识的资源接收数据
            mapSebClient.Credentials = CredentialCache.
DefaultCredentials; // 获取或设置用于对向 Internet 资源的请
//求进行身份验证的网络凭据。
            string xmlStr = "";
            while (xmlStr == "")
            {
           Encoding enc = Encoding.GetEncoding("utf-8");
// 如果是乱码就改成 GB2312/utf-8
          Byte [] pageData = mapSebClient.DownloadData
(POISearchUrl); // 从资源下载数据并返回字节数组。
          xmlStr = enc.GetString(pageData); // 输出字符串
//(HTML 代码),ContentHtml 为 Multiline 模式的 TextBox 控件
            }
            XmlDocument xmlDoc = new XmlDocument();
            xmlDoc.LoadXml(xmlStr);
      //使用 xpath 表达式选择文档中所有的 result 子节点
            XmlNodeList NodeList = xmlDoc.SelectNodes("
PlaceSearchResponse/results/result");
            if (NodeList ! = null)
            {
               try
               {
                  SQLiteDB sqliteDbObj = new SQLiteDB();
                  sqliteDbObj.openConnection("");
                  foreach (XmlNode stNode in NodeList)
                  {
          //通过 Attributes 获得属性名字为 name 的属性
   string name = stNode.SelectNodes("name")[0].InnerText;
  string lat = stNode.SelectNodes("location/lat")[0].InnerText;
 string lng = stNode.SelectNodes("location/lng")[0].InnerText;
                     string trainStopName = System.Text.
RegularExpressions.Regex.Replace(name, @"市", "");
   string sqlStr = "update CapitalOfprovince set  lat = " + lat +
",lng=" + lng + " where  cityName = '" + trainStopName + "'";
          sqliteDbObj.ExecuteSQLiteSQL(sqlStr);
                  }
               }
               catch (Exception e)
               {
                  Console.Write(e.Message);
               }
            }
```

```
        }
        private void createStopLocationInfo (string region,
string lat, string lng)
        {
            string POISearchUrl = "http://api.map.baidu.
com/place/search?&query=火车站 &location=" + lat + "," +
lng + "&radius=20000&output=xml&key=46b68150f22408ef
4e4dcf3281ecb400"; //需要获取源代码的
            WebClient mapSebClient = new WebClient();
//创建 WebClient 实例提供向 URI 标识的资源发送数据和从
//URI 标识的资源接收数据
            mapSebClient.Credentials = CredentialCache.
DefaultCredentials; // 获取或设置用于对向 Internet 资源的请
//求进行身份验证的网络凭据。
            string xmlStr = "";
            while (xmlStr == "")
            {
               Encoding enc = Encoding.GetEncoding("utf-
8"); // 如果是乱码就改成 GB2312/utf-8
          Byte [] pageData = mapSebClient.DownloadData
(POISearchUrl); // 从资源下载数据并返回字节数组。
          xmlStr = enc.GetString(pageData); // 输出字符串
//(HTML 代码),ContentHtml 为 Multiline 模式的 TextBox 控件
            }
            XmlDocument xmlDoc = new XmlDocument();
            xmlDoc.LoadXml(xmlStr);
      //使用 xpath 表达式选择文档中所有的 result 子节点
            XmlNodeList NodeList = xmlDoc.SelectNodes("
PlaceSearchResponse/results/result");
            if (NodeList ! = null)
            {
               try
               {
                  SQLiteDB sqliteDbObj = new SQLiteDB();
                  sqliteDbObj.openConnection("");
                  foreach (XmlNode stNode in NodeList)
                  {
          //通过 Attributes 获得属性名字为 name 的属性
   string name = stNode.SelectNodes("name")[0].InnerText;
 string lat1 = stNode.SelectNodes("location/lat")[0].InnerText;
string lng1 = stNode.SelectNodes("location/lng")[0].InnerText;
                     string trainStopName = System.Text.
RegularExpressions.Regex.Replace(name, @"站", "");
          string sqlStr = "insert into trainStopLocationCity
(trainStopName,cityName,lat,lng) values('" + trainStopName
+ "','" + region + "'," + lat1 + "," + lng1 + ")";
          sqliteDbObj.ExecuteSQLiteSQL(sqlStr);
                  }
               }
               catch (Exception e)
               {
                  Console.Write(e.Message);
```

```
            }
        }
    }
}
```

得到坐标位置和站点位置“包含于”关系的样例数据，如图 5 所示。

| stopName | LocationCity |
|---|---|
| 上海 | 上海 |
| 上海南 | 上海 |
| 上海虹桥 | 上海 |
| 上海西 | 上海 |
| 松江 | 上海 |
| 北京 | 北京 |
| 北京南 | 北京 |
| 北京西 | 北京 |
| 黄村 | 北京 |
| 仙林 | 南京 |
| 南京 | 南京 |
| 南京南 | 南京 |
| 余杭 | 杭州 |
| 杭州 | 杭州 |
| 杭州南 | 杭州 |
| 武昌 | 武汉 |
| 武汉 | 武汉 |
| 沈阳 | 沈阳 |
| 蒲井子 | 沈阳 |
| 苏家屯 | 沈阳 |
| 西安 | 西安 |
| 西安北 | 西安 |
| 西安南 | 西安 |

图 5 获取地理定位信息的样例数据

## 5 使用 Jena 创建本体的类、属性和个体

一般来说，可以在 Protege 这样的编辑器里构建了本体，但用程序操作本体是很有必要的，因为在很多情况下如要从关系数据库中自动生成大量的本体，靠人手通过 Protege 创建所有本体是不现实的。本应用程序里使用 Jena 创建 OWL 格式的本体。

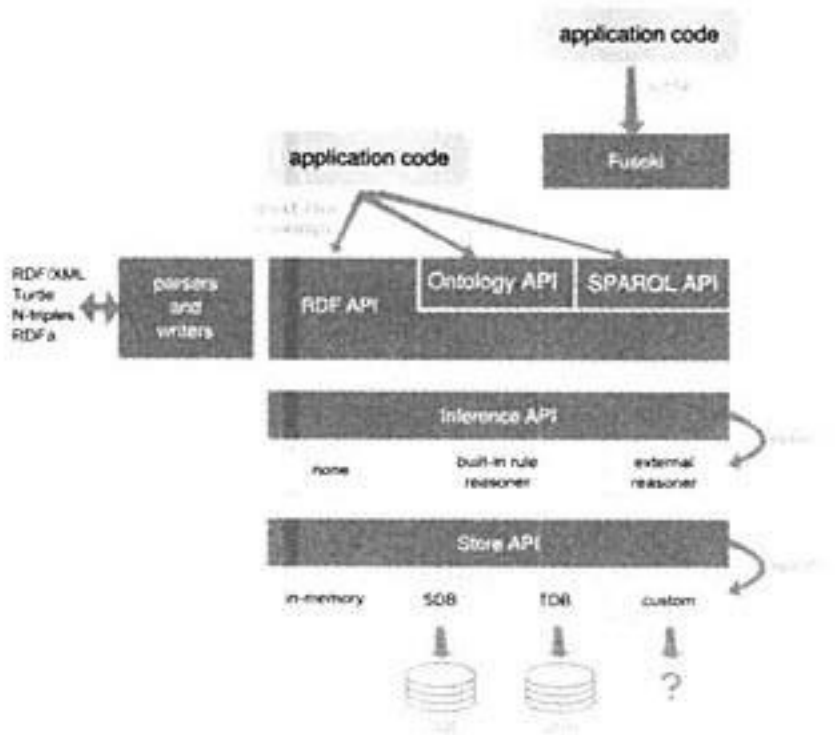



图 6 Jena 结构图

Jena 最基本的使用是处理 RDF (S)，但毕竟 OWL 已经成为 W3C 的推荐标准，因此程序将使用 com.hp.hpl.jena.ontology 接口根据 SQLite 表的数据来创建本体。OWL 核心是围绕三元组的，即在很多资料中看到的陈述 (Statement)，它的组成为：Statement= (Subject，Property，Object)，其中 Subject 为主体，Property 成为属性，Object 成为客体。Individual 称为个体。它是 Subject、Object 的一个实例，例如在 Statement= (Animals，Eat，Plants) 陈述中 sheep 是 Animals 的一个 Individual，grass 是 Plants 的一个 Individual。一切的主体、客体、个体、类、属性等都可以称为资源 (Resource)。在编程中 OWL 常用的是 OntModel，Ontclass，OntProperty，Individual。OntClass 可以用来创建主体、客体，OntPropety 用来创建属性，Individual 用来创建个体，Jena 结构如图 6 所示。

因此，对应于关系数据库-本体，在本应用程序中对应关系是：数据表-class (类)、字段-属性 (Property)、记录-个体 (Individual)。

下面介绍代码具体实现：

```
using jena;
using com.hp.hpl.jena.db;
using com.hp.hpl.jena.rdf.model;
using com.hp.hpl.jena.query;
using ikvm.lang;
using com.microsoft.sqlserver.jdbc;
using System.Xml;  //在.net 中操作 OWL 或 RDF 还是要
//引入 xml 库
using System.Xml.XPath;
using ikvm.io;
using com.hp.hpl.jena.ontology;
using com.hp.hpl.jena.reasoner;
using com.hp.hpl.jena.reasoner.rulesys;
using com.hp.hpl.jena.vocabulary;
using System.Collections;
//…
  //本体命名空间类
  public class owlPreFix
  {
     private string _preFixShortStr;
     private string _preFixStr;
     public string preFixShortStr
     {
        get { return _preFixShortStr; }
        set { _preFixShortStr = value; }
     }
     public string preFixStr
     {
        get { return _preFixStr; }
        set { _preFixStr = value; }
     }
  }
  //创建本体类
  class createOwlByjena
  {
     private OntModel _model;
     private string _preFixShortStr;
     private string _preFixStr;
     public OntModel model
     {
```

FOLLOW MASTER PROGRAM

```
        get { return _model; }
        set { _model = value; }
    }
    //创建本体对象
    public createOwlByjena(OntModel model)
    {
        if (model ! = null)
            _model = model;
        if (_model == null)
            _model = ModelFactory.createOntologyModel();
            }
            //设置本体命名空间
             public void setOwlPrefix (string preFixShortStr,
string preFixStr)
            {
                model.setNsPrefix(preFixShortStr, preFixStr);
                model.setNsPrefix("base", preFixStr);
                _preFixShortStr = preFixShortStr;
                _preFixStr = preFixStr;
            }
            //创建本体中的类
          public OntClass createOwlClass (string className,
ArrayList DataColumnAL)
            {
                className = _preFixStr + className;
          OntClass trainStop = model.createClass(className);
                for (int i = 0; i < DataColumnAL.Count; i++)
                {
                    createOwlProperty (DataColumnAL [i + 1].
ToString(), DataColumnAL[i].ToString());
                    i++;
                }
                return trainStop;
            }
            //创建本体中类的属性
                public OntProperty createOwlProperty (string
propertyName, string type)
            {
                propertyName = _preFixStr + propertyName;
                OntProperty trainStopProperty = null;
                if (type == "1")
            trainStopProperty = model.createDatatypeProperty
(propertyName);
                if (type == "0")
                trainStopProperty = model.createObjectProperty
(propertyName);
                return trainStopProperty;
            }
            //创建本体中个体
              public Individual createOwlIndividual (string
IndividualName, OntClass IndividualOfClass)
            {
                return IndividualOfClass.createIndividual(IndividualName);
            }
            //创建本体中个体的数据属性值
       public Resource setOwlIndividualDatatypePropertyValue
(string OntDataPropertyName, string OntDataPropertyValue,
Individual trainStopIndividual)
            {
      com.hp.hpl.jena.ontology.DatatypeProperty OntProperty =
trainStopIndividual.getOntModel().getDatatypeProperty(Ont
DataPropertyName);
            return trainStopIndividual.addProperty (OntProperty,
OntDataPropertyValue);
            }
            //创建本体中个体的对象属性值
          public Resource setOwlIndividualObjectPropertyValue
(string OntObjectPropertyName, string OntObjectProperty
Value, Individual trainStopIndividual)
            {
           com.hp.hpl.jena.ontology.ObjectProperty OntProperty
=trainStopIndividual.getOntModel().getObjectProperty(OntObj
ectPropertyName);
            return trainStopIndividual.addProperty (OntProperty,
OntObjectPropertyValue);
            }
        }
```

## 6 结语

介绍了基于 SQLite 数据引擎、地图 Web 服务和 C# 环境中的 Jena，实现地理定位信息的网络获取和本体的类、属性等的构建。这一部分功能为利用语义网技术实现铁路交通的地理语义查询的实现进行了数据预处理和基本操作的准备工作。从本篇实现的功能可以看出，SQLite 的执行效率高、地图 Web 服务能够方便提供有用的地理空间定位信息，而使用 IKVM，可以使基于 Java 的开源程序很方便地集成到 C# 中使用。

## 参考文献


[1] 董志. C# 集成 Google Map API 进行地理空间的定位 [J]. 电脑编程技巧与维护，2011 (19)：47-53.

[2] Baader F, Horrocks I, Sattler U. Description Logics as Ontology Languages for the Semantic Web [ M] //Lecture Notes in Artificial Intelligence. [S. l.]: Springer, 2005.

[3] http://www.ibm.com/developerworks/cn/java/j-jena/.

[4] http://www.sqlite.org/cintro.html.

[5] http://developer.baidu.com/map/.

[6] Smith, Michael K.; Chris Welty, Deborah L. McGuinness. OWL Ontology Web Language Guide. W3C. 2004-02-10 [2008-07-15].


（收稿日期：2012-01-24）

# 用 VC++ 实现对话框的界面设计

蔡智明　杨秋瑾

**摘　要**：阐述了对话框界面设计的基本概念和原理，详细说明了以绘制位图技术为背景的对话框绘制界面技术，重点介绍了对话框界面设计方案的设计与实现，主要包括对话框本身界面的重绘和按钮控件的重写以及应用。

**关键词**：对话框；重绘；用户界面；位图；VC++语言

## 1　引言

应用程序是用户与应用程序之间交互的桥梁和媒介，用户界面是应用程序中最重要的组成部分，也是最为直观的视觉体现。对用户而言，界面就是应用程序，界面设计的好坏会直接影响应用程序的可用性，从而影响用户的体验。

在软件开发过程中，对界面的设计都是一项很重要的技术，如今的应用软件界面可谓是“丰富多彩、美丽绝伦”，如大家熟悉的 360 安全卫士、腾讯 QQ 聊天软件、Visual C++ 编程词典软件等，都是非常不同于普通的对话框应用程序，因为他们的界面都是重新绘制过的，从而实现了漂亮、易用的用户体验。鉴于 VC++编程技术，下面将通过对话框的重新绘制来达到自定义的界面效果。

## 2　概述

实现对话框界面的设计，即对对话框重新自定义绘制，主要包括标题栏的重绘、对话框边框的重绘、对话框背景重绘、以及最小化按钮、最大化按钮和关闭按钮等的重绘实现。

## 3　编程平台与技术

编程平台使用 Microsoft Visual Studio 2008 集成开发环境，编程技术采用 Visual C++ 编程技术，以及相关的开发软件如 Photoshop CS5 等。

## 4　方案分析

在对话框重绘中，使用的主要技术有两个，一个是绘制对话框的背景位图，在对话框大小改变时能够输出位图，使位图能够适应对话框的大小。另一个是在对话框的指定区域输出位图。

### 4.1　绘制对话框的背景位图

绘制对话框背景位图本文采用的是处理对话框的 WM_PAINT 消息，该消息初始化时候对对话框进行绘制，从而绘制背景位图。绘制背景位图的主要代码如下：

```
CRect  rect;
 CPaintDC  dc(this);
 GetClientRect(&rect); //获取客户区
//设置对话框背景颜色
 dc.FillSolidRect(rect,RGB(14,94,157));  //设置为窗口背景
```

### 4.2　在指定的区域中输出位图

为了能够在指定的区域中输出位图，需要使用设备上下文 CDC 类的 StretchBlt 方法。由于需要在窗口的非客户区域绘制位图，因此需要使用 CWindowDC 类的 StretchBlt 方法，CWindowDC 类派生于 CDC 类，它提供了在窗口非客户区域绘制位图的功能。该方法数从源矩形中复制一个位图到目标矩形，必要时按目前目标设备设置的模式进行图像的拉伸或压缩。输出位图的主要实现代码如下：

```
CRect winRC;
CDC* pDC=GetWindowDC();//获取窗口设备上下文
CDC memDC;
memDC.CreateCompatibleDC(pDC);//创建兼容内存位图
BITMAPINFO bmpInfo;
CBitmap bmp;  //定义位图对象
GetWindowRect(&winRC);
bmp.LoadBitmap(nID);//加载位图
  bmp.GetObject(sizeof(BITMAPINFO),&bmpInfo);//获取
//位图信息
int nBmpCX = bmpInfo.bmiHeader.biWidth;//获取位图宽度
int nBmpCY = bmpInfo.bmiHeader.biHeight;//获取位图高度
memDC.SelectObject(bmp);//选中位图对象
pDC->StretchBlt(x,y,w,h,
      &memDC,0,0,nBmpCX,nBmpCY,SRCCOPY);//在窗
//口中绘制位图
bmp.DeleteObject();//释放位图对象
ReleaseDC(pDC);//释放 DC
```

## 5　设计与实现

### 5.1　方案设计

对界面的整体重绘包括两部分：一部分是对话框自身的

重绘；二是对话框控件的重绘。这里主要介绍按钮控件的重绘。

## 5.2 对话框绘制

在对话框重绘的设计与实现过程中，一般需要绘制的对话框区域主要有标题部分、边框部分和客户区部分。具体的区域划分如图 1 所示。

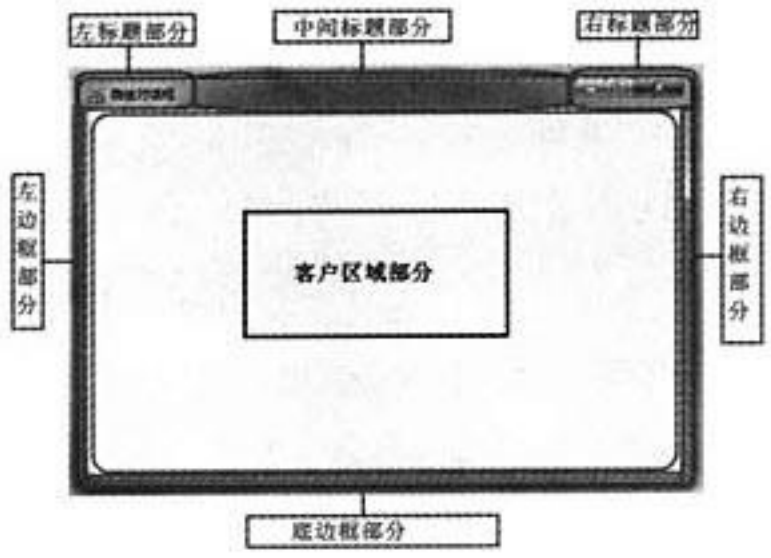



**图 1 对话框绘制区域图**

既然要对多个区域进行位图显示输出，所以先封装一个 bmp 位图显示输出函数如下：

```
void CCTestDlg::ShowBmp(int x,int y,int w,int h,int nID)
{//nID 表示位图资源的 ID
    CRect winRC;
    CDC* pDC=GetWindowDC();
    CDC memDC;
    memDC.CreateCompatibleDC(pDC);
    BITMAPINFO bmpInfo;
    CBitmap bmp;
    GetWindowRect(&winRC);
    bmp.LoadBitmap(nID);
    bmp.GetObject(sizeof(BITMAPINFO),&bmpInfo);
    int nBmpCX = bmpInfo.bmiHeader.biWidth;
    int nBmpCY = bmpInfo.bmiHeader.biHeight;
    memDC.SelectObject(bmp);
    pDC->StretchBlt(x,y,w,h,
        &memDC,0,0,nBmpCX,nBmpCY,SRCCOPY);//在窗
//口中绘制位图
    bmp.DeleteObject();
    ReleaseDC(pDC);
}
```

(1) 对各个区域进行位图输出重绘。由于标题栏以及边框主要都是非客户区域绘制，因此应该在 WM_NCPAINT 消息中绘制。当然得先通过添加资源的方式将所用到的 bmp 位图资源导入到项目中。

首先定义一些常量值，表示对话框各个组成区域部分。代码如下：

```
#define LEFTTITLE 1//左标题
#define MIDTITLE 1//中间标题
#define RIGHTTITLE 1//右标题
#define MINBUTTON 1//右标题
#define MAXBUTTON 1//右标题
#define CLOSEBUTTON 1//右标题
#define APPICON 1 //程序 icon 图标
#define LEFTBAR 1 //左边框
#define RIGHTBAR 1 //右边边框
#define BOTTOMBAR 1 //底边框
```

在 WM_NCPAINT 消息对于的 方法 OnNcPaint ()中调用对话框绘制方法 SetFace ()。该方法的功能就是绘制对话框各个区域的位图。主要代码如下：

```
void CCTestDlg::SetFact()
{
    // TODO: 在此添加控件通知处理程序代码
                    int nFrameCY = GetSystemMetrics
(SM_CYFIXEDFRAME);//获取对话框边框的高度
                    int nFrameCX = GetSystemMetrics
(SM_CXDLGFRAME);//获取对话边框的宽度
    int m_nBorderCY;
    int m_nBorderCX;
    int m_nTitleBarCY ;
    int m_nTitleBarCX;
    if(GetStyle()&WS_BORDER)//获取对话框是否有边框
    {
                    m_nBorderCY = GetSystemMetrics
(SM_CYBORDER) + nFrameCY;
                    m_nBorderCX = GetSystemMetrics
(SM_CXBORDER) +nFrameCX;
    }
    else
    {
        m_nBorderCX = nFrameCX;
        m_nBorderCY = nFrameCY;
    }
    m_nTitleBarCY = GetSystemMetrics (SM_CYCAPTION)
+ m_nBorderCY;//计算标题栏高度
    m_nTitleBarCX =m_nBorderCX;
    CRect winRect,factRect;
    GetWindowRect(&winRect); //获取窗口区域
    factRect.CopyRect (CRect (0,0,winRect.Width (),winRect.
Height()));
    CWindowDC windowsDC(this);//获取窗口设备上下文
//获取整个 MFC 窗口的高度和宽度
int winCX = winRect.Width();
int winCY = winRect.Height();
if(LEFTTITLE)
{//绘制对话框左标题栏位图
ShowBmp(0,0,100,m_nTitleBarCY,IDB_RIGHTTITLE);
}
if(RIGHTTITLE)
{//绘制对话框右标题栏位图
        ShowBmp  (winCX-100,0,100,m_nTitleBarCY,
```

```
IDB_RIGHTTITLE);
        }
        if(MIDTITLE)
        {//绘制对话框中标题栏位图
ShowBmp(100,0,winCX-200,m_nTitleBarCY,IDB_MIDTITLE);
        }
        if(LEFTBAR)
{//绘制对话框左边框位图
ShowBmp(0,m_nTitleBarCY,m_nBorderCX,factRect.Height()-
m_nBorderCY,IDB_LEFTBAR);
        }
        if(BOTTOMBAR)
         {//绘制对话框底边框位图
ShowBmp (m_nBorderCX, winCY - m_nBorderCX, winCX -
2*m_nBorderCX,m_nBorderCX,IDB_BOTTOMBAR);
        }
        if(RIGHTBAR)
         {//绘制对话框左边框位图
ShowBmp(winCX-m_nBorderCX,m_nTitleBarCY,m_nBorder
CX,factRect.Height()-m_nBorderCY,IDB_RIGHTBAR);
        }
        if(MINBUTTON)
        {//给对话框绘制最小化按钮
              ShowBmp (winCX -3 -24 -3 -24 -3 -24,1,24,24,
IDB_MINBUTTON);
        }
        if(MAXBUTTON)
        {//给对话框绘制最大化按钮
                    ShowBmp (winCX -3 -24 -3 -24,1,24,24,
IDB_MAXBUTTON);
        }
        if(CLOSEBUTTON)
        {//给对话框绘制关闭按钮
        ShowBmp(winCX-3-24,1,24,24,IDB_CLOSEBUTTON);
        }
        ReleaseDC(&windowsDC);
        DrawTitleBarText();//输出标题栏文本
}
```

上面代码中最后的绘制对话框标题文本的方法DrawTitleBar Text () 的主要代码如下：

```
        CString strTitle ="自绘窗口标题栏和边框";
         CDC* pDC= GetWindowDC();//获取窗口设备上下文
pDC->SetBkMode(TRANSPARENT);//设置透明的背景模式
        pDC->SetTextColor(RGB(255,255,255));//设置文本颜色
        pDC->SetTextAlign(TA_CENTER);//设置文本对齐方式
        CRect rect;
        GetClientRect(&rect);//获取窗口客户区域
          CSize szText = pDC->GetTextExtent(strTitle);//获取文
//本高度
        pDC->TextOut(rect.Width()/2,3,strTitle,20);//在窗口中输
//出文本
ReleaseDC(pDC);//释放窗口设备上下文
```

绘制后的效果如图 2 所示。

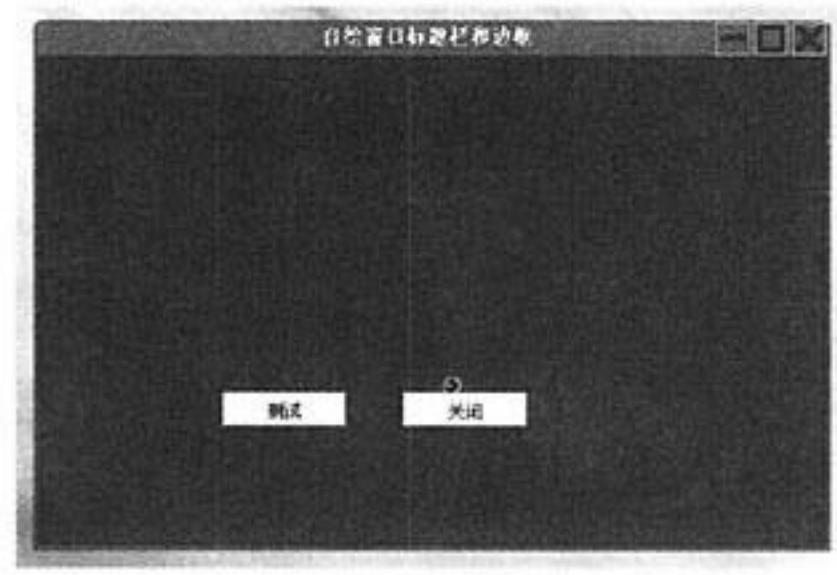



**图 2 对话框绘制效果图**

在完成对话框相应区域的位图后，并没有完成任务，还需要处理标题栏按钮的热点效果，以及按钮的单击事件。首先得处理鼠标在非客户区域移动时的事件，即WM_NCMOUSEMOVE消息，在其消息处理函数中判断当前的鼠标点是否位于标题栏的按钮区域，如果是则设置按钮的热点效果，并且记录当前的按钮状态，及鼠标点在哪个按钮上。同样，处理对话框非客户区域的单击事件，即 WM_NCLBUTTONDOWN 消息，在其消息处理函数中完成单击事件操作。主要代码如下：

```
void CC TestDlg::OnNcMouseMove (UINT nHitTest, CPoint
point)
{
        // TODO: 在此添加消息处理程序代码和/或调用默认值
        CRect minRC,maxRC,closeRC,winRC;
        GetWindowRect(&winRC);
        closeRC.CopyRect (CRect (winRC.Width () -27,1,winRC.
Width()-27+24,1+24));
        maxRC.CopyRect(CRect(winRC.Width()-27*2,1,winRC.
Width()-27*2+24,1+24));
        minRC.CopyRect(CRect(winRC.Width()-27*3,1,winRC.
Width()-27*3+24,1+24));
          point.Offset(-winRC.left,-winRC.top);//由于 point 为屏
//幕坐标,这里将其转换为窗口坐标
         if(closeRC.PtInRect(point)) //判断鼠标是否在关闭按钮区
//域上
        {
                    ShowBmp (winRC.Width () -3 -24,1,24,24,
IDB_CLOSEBUTTON2);
        }
          else if(maxRC.PtInRect(point)) //判断鼠标是否在最大化
//按钮区域上
        {
ShowBmp(winRC.Width()-27*2,1,24,24,IDB_MAXBUTTON2);
        }
        else if(minRC.PtInRect(point)) //判断鼠标是否在最小化按
//钮区域上
        {
```

```
        ShowBmp    (winRC.Width    ()   -27*3,1,24,24,
IDB_MINBUTTON2);
    }
    else//鼠标没有在标题栏的按钮区域上
    {
        ShowBmp   (winRC.Width   () -3 -24 -54,1,24,24,
IDB_MINBUTTON);
        ShowBmp   (winRC.Width   () -3 -24 -27,1,24,24,
IDB_MAXBUTTON);
        ShowBmp    (winRC.Width    ()   -3   -24,1,24,24,
IDB_CLOSEBUTTON);
    }
}
```

添加热点后的效果如图 3 所示。

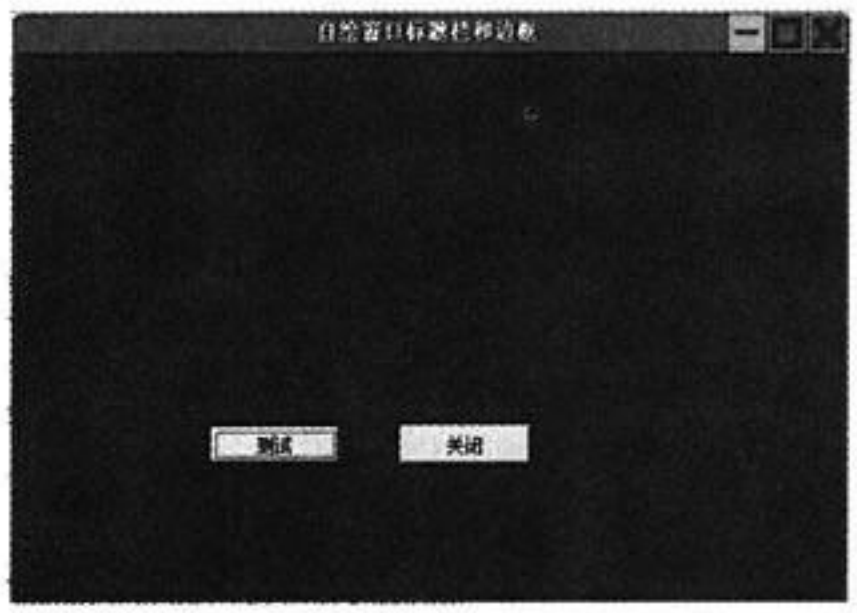



图 3　绘制对话框之热点效果图

## 5.3　按钮控件重绘

在 MFC 下编程，很多时候对于标准的按钮控件不是很满意，想要弄得美观些。这就需要按钮重绘。重绘按钮一般的实现方法就是重写 CButton 类。

首先给工程添加一个自绘按钮类 MyDrawButton，基类为 CButton。要想让按钮具备自绘功能，就要为按钮添加 BS_OWNERDRAW 属性。为类 CButton 重载 PreSubclassWindow 虚函数。在该函数中添加如下一行代码：

```
SetButtonStyle(GetButtonStyle() | BS_OWNERDRAW);
```

当按钮控件具有了自绘功能之后，每次控件状态改变都会触发 DrawItem 函数，在该函数中来绘制按钮的形态外观，所以第二步就要重载 DrawItem 虚函数。在这个函数中就可以自由发挥了，比如绘制背景、底色、按钮标题,绘制文本字体样式等。

一般都会为按钮定义几种不同状态时的外观，比如光标滑过时的状态、按钮按下时的状态、按钮禁用时的状态，以及按钮的正常状态等。这就要为新的按钮添加几种重要的消息响应。比如 WM_MOUSELEAVE 消息，WM_MOUSEHOVER 消息和 WM_MOUSEMOVE 消息等，值得一提的是前两个消息的响应函数需要自己手动添加，微软提供了一个TrackMouseEvent 函数在光标离开一个窗口时投递 WM_MOUSELEAVE 消息，光标滑过窗口时投递 WM_MOUSEHOVER 消息。一般来说可以在 WM_MOUSEMOVE 消息响应函数中调用 TrackMouseEvent 函数来投递 WM_MOUSELEAVE 消息和 WM_MOUSEHOVER 消息。然后在 WM_MOUSELEAVE 消息的响应函数中标记“光标已经离开按钮”，然后调用 InvalidateRect 函数让按钮重绘。在 WM_MOUSEHOVER 消息的响应函数中标记“光标正在按钮上方”，并调用 InvalidateRect 函数让按钮重绘。

重绘按钮分为 3 个部分。

(1) 绘制按钮背景样式，即绘制背景 bmp 位图，使得按钮具有自定义的样式，同时在绘制按钮背景的输出位图时采用 TransparentBlt()函数，该函数的作用是使窗体上显示位图的背景与窗体背景色融为一体，不仅可以显示按钮 bmp 位图样式，而且还可以使背景透明。

(2) 绘制按钮上的文本。主要绘制按钮上文本的样式，包括字体大小、字体样式、字体颜色等属性。

(3) 实现不同状态下的按钮的外观样式，主要包括 WM_MOUSEMOVE 和 WM_MOUSELEAVE 两个消息的消息处理函数。分别实现鼠标在按钮区域上和不在按钮区域上的状态。为了标记鼠标移动到按钮区域内停留，需要用到一个定时器来标记鼠标是否还在按钮区域内停留。在WM_MOUSEMOVE 内启动定时器，触发 WM_MOUSELEAVE 消息时结束定时器即销毁定时器。定时器的主要代码如下：

```
void MyDrawButton::OnTimer(UINT_PTR nIDEvent)
{
    // TODO: 在此添加消息处理程序代码和/或调用默认值
    if(nIDEvent ! = 24)
        return;
    CPoint    point;
    CRect     rect;
    GetWindowRect(&rect);
    GetCursorPos(&point);
    // 如果鼠标离开按钮区域,重绘按钮
    if (! rect.PtInRect(point) && m_bMove)
    {
        KillTimer (24);
        m_DrawState=ST_MOVEOUT;
        m_bMove=FALSE;
        Draw();
    }
    CButton::OnTimer(nIDEvent);
}
```

重绘按钮类 MyDrawButton 的主要实现代码如下：

定义的一些重绘用到的变量：

```
#define ST_MOVEIN    0//绘制状态—在按钮区域上
#define ST_MOVEOUT   1 //绘制状态—不在按钮区域上
int m_DrawState;//绘制状态
 int m_nBmpID;//当前显示的背景 bmp 位图的资源 ID
```

```
    bool m_bMove;//鼠标是否进入按钮区域
    COLORREF m_clText;//当前文本颜色
     COLORREF m_clActiveText;//鼠标进入按钮区域时文本
//颜色
    COLORREF m_clNormalText;//鼠标离开按钮区域时文本
//颜色
```

消息处理函数和定义的函数：

```
void MyDrawButton::PreSubclassWindow()
{
    SetButtonStyle(GetButtonStyle() | BS_OWNERDRAW);
}
void    MyDrawButton::DrawItem    (LPDRAWITEMSTRUCT
lpDrawItemStruct)
{
      Draw();//绘制按钮
}
void MyDrawButton::Draw()//绘制按钮
{
DrawBackground();//绘制按钮 bmp 位图,并使背景透明化
      DrawText();//绘制按钮上的文本
}
void MyDrawButton::DrawText()
{//绘制按钮上的文本的字体大小、样式等
      CString itemString;
      CRect  clientRect;
      CClientDC  dc(this);
      GetClientRect(&clientRect);
      GetWindowText(itemString);
      if(itemString)
      {
            CSize size=dc.GetTextExtent (itemString);//获得所
//选字体中指定字符串的高度和宽度
            int rectwidth=clientRect.Width();
            int rectheight=clientRect.Height();
            int textwidth=size.cx ;
            int textheight=size.cy ;
            int x,y; // 文本的位置
            // 计算文本的输出位置
            x=(rectwidth-textwidth)/2;//水平居中
            y=(rectheight-textheight)/2;//垂直居中
            switch(m_DrawState)
            {
            case ST_MOVEIN://鼠标进入按钮区域
                  m_clText=m_clActiveText;
                  break;
            case ST_MOVEOUT://鼠标离开按钮区域
                  m_clText=m_clNormalText;
                  break;
            default:
                  m_clText=m_clNormalText;
                  break;
            }
            dc.SetTextColor(m_clText);
            dc.SetBkMode(TRANSPARENT);
            CFont *font ;
            font =new CFont();
             int fontSize = 14;font->CreateFont(fontSize,0,0,0,
FW_BOLD,FALSE,FALSE,0,ANSI_CHARSET,
OUT_DEFAULT_PRECIS,CLIP_DEFAULT_PRECIS,
DEFAULT_QUALITY,FF_SWISS,_T("宋体"));
            dc.SelectObject(font);
            dc.TextOut (x,y,itemString);
      }
}
void MyDrawButton::SetBkBmp(int nBmpID)
{//设置按钮 bmp 位图样式
      m_nBmpID = nBmpID;
}
void MyDrawButton::DrawBackground()
{//绘制按钮 bmp 位图,并使背景透明化
      CRect winRC;
      CDC* pDC=GetWindowDC();
      CDC memDC;
      memDC.CreateCompatibleDC(pDC);
      BITMAPINFO bmpInfo;
      CBitmap bmp;
      GetWindowRect(&winRC);
      bmp.LoadBitmap(m_nBmpID);
      bmp.GetObject(sizeof(BITMAPINFO),&bmpInfo);
      int nBmpCX = bmpInfo.bmiHeader.biWidth;
      int nBmpCY = bmpInfo.bmiHeader.biHeight;
      memDC.SelectObject(bmp);
pDC->TransparentBlt(0,0,nBmpCX,nBmpCY,&memDC,0,0,
          nBmpCX,nBmpCY,RGB(14,94,157));//在窗口中绘制
//位图,RGB(14,94,157)是透明色
      bmp.DeleteObject();
      ReleaseDC(pDC);
}
```

到此，按钮的自定义重绘完成了，接下来就可以使用自己重绘的按钮类 MyDrawButton 了。首先往对话框中添加一个按钮控件，假设它的 ID 值为 IDC_TEST。进入类向导（Class Wizard）的成员变量属性页，为 IDC_ TEST 添加一个变量 m_testButton。如下：

```
MyDrawButton m_testButton;
```

然后就可以调用 MyDrawButton 的方法来设置按钮的样式了。如下：

```
 m_testButton.SetBkBmp(IDB_TEST);//IDB_TEST 为所设置的
//bmp 位图资源 ID。
```

到现在为止，按钮类的重绘完成了，可以随意定义自己喜欢的样式的按钮了。带有自定义按钮的对话框重绘效果如图 4

（下转第 20 页）

# VC++ 编程实现多模式近似匹配

张 研 韩 露

**摘 要**：提出了一种实现多模式近似匹配的 BKDR-BPM 算法，分析了算法的设计思路及关键步骤，并用 VC++编程实现。

**关键词**：近似匹配；多模式；算法

## 1 引言

目前，近似字符串匹配在入侵检测、文件检索、文本过滤、信息查询等许多领域有着广泛的应用，是当今计算机算法研究的重要课题之一。虽然单模式近似匹配算法的研究已经日趋成熟，但是，多模式近似匹配算法仍存在缺少支持汉字等大字符集的现状。文中在研究多模式精确匹配算法及单模式近似匹配的基础上，提出了一种能够实现大字符集多模式近似匹配的 BKDR-BPM 算法，并给出了算法的 C++语言实现。

## 2 BKDR-BPM 算法

### 2.1 近似匹配的定义

近似字符串匹配指的是给定一个文本 T [1:n]，n 是文本长度，一个模式 P [1:m]，m 是模式的长度，以及容许的最大误差 k (k<m)，找出文本 T 中满足误差小于 k 的所有字符串 S，即 ed (S,P) <=k，其中 ed (S,P) 表示字符串 S 和 P 的距离。

### 2.2 算法设计

受经典的多模式匹配算法—WM 算法[1]的启发，多模式近似匹配算法分两步进行，第一步是对多个模式字符串求出 HASH 值，构造 HASH 表，为了最大程度地分散链表的入口地址，采用 BKDRHash 算法[2]进行散列处理，当模式数量大时，BKDRHash 算法的碰撞率较低，性能较好。在扫描文本过程中，一旦遇到 HASH 值相等的字符串，则进行第二步处理；第二步是进行单个模式的近似匹配，单模式近似匹配采用动态生成矩阵与位向量相结合的 BPM 算法[3]，因此多模式近似匹配算法命名为 BKDR -BPM 算法。图 1 是该算法的流程图。

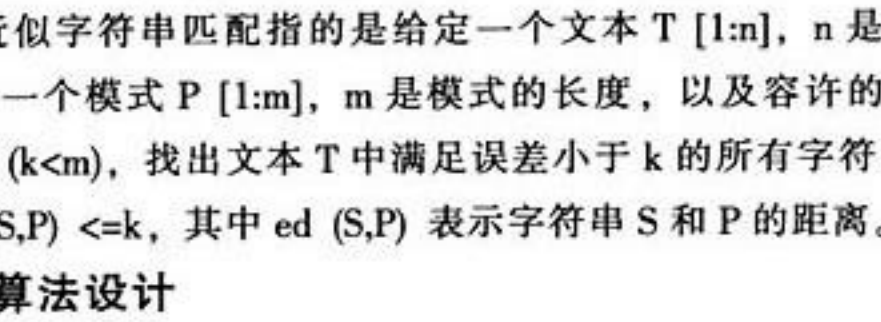



图 1 BKDR -BPM 算法流程

BKDR-BPM 算法首先对输入的文本按块计算 HASH 值，块的大小通常设置为 4 个字节，计算 HASH 值采用 BKDRHash 算法，根据求得的 HASH 值查 HASH 表，如 HASH 表的链指针不为空，则说明有可能发生匹配，需要进行单模式的近似匹配。在进行单模式近似匹配时采用 BPM 算法。以模式集合“科学技术”和“理论研究”为例，假设用 BKDRHash 算法求得的 HASH 值同为 h，则这两个词共用同一个 HASH 表入口地址，为它们创建的结点以链表方式连接，当扫描文本时遇到 HASH 值等于 h 的字符串时，则调用 BPM 算法分别与 HASH [h] 指向的链表结点进行近似匹配，如匹配成功则保存结果。HASH 表如图 2 所示。

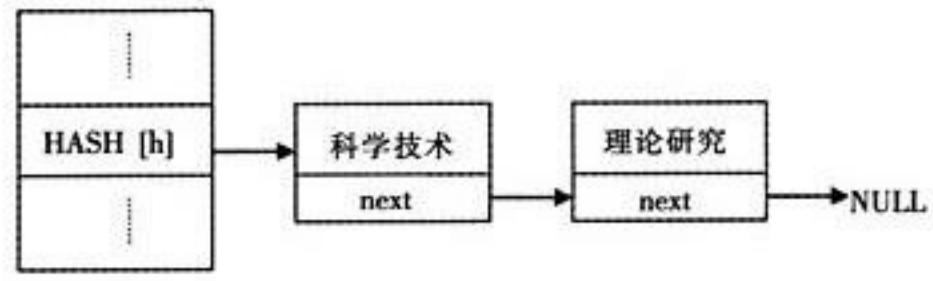



图 2 HASH 表的结构

### 2.3 算法的数据结构

算法的数据结构主要是 HASH 表，HASH 表的结点可用下面的结构体描述，其中，pattern [64] 表示模式，即字符串，每个模式长度为 64 字节，对于英文这样的小字符集语言（一个字母占用一个字节）可以支持 64 个字母，对于中文这样的大字符集语言（一个汉字占用两个字节）可以支持 32 个汉字。一个字节最多表示 256 种字符，所以用数组 Peq [256] 表示，变量定义为 __int64 类型是因为当前计算机字长通常不超过 64 位。_SIZE_OF_TABLE 是 HASH 表的大小，在本算法中定义为 32K，即 215=32768，因此，BKDRHash 算法求出的 HASH 值需用 15 位表示：

```
struct PatNode
{
    char pattern[64];
    __int64 Req[256];
    struct PatNode *next;
};
struct PatNode *Hash[_SIZE_OF_TABLE];
```

## 3 BKDR-BPM 算法的 C++实现

下文用 C++语言实现 BKDR -BPM 算法，并用 VC++6.0 作为开发平台，设计了一个基于对话框的多模式近似匹配的工程。

### 3.1 CHashBpm 类的设计与实现

工程构造一个 CHashBpm 类用于实现 BKDR-BPM 算法，类的定义代码如下：

```
class CHashBpm
{
public:
    void Release(void); //释放资源
     int AddPattern(char *pat); //增加待匹配的模式串
     int Match(const char *text, int distance); //进行匹配
     CHashBpm();   //构造函数
     virtual ~CHashBpm();      //析构函数
#define _SIZE_OF_TABLE 32*1024  //定义字符集,以字节为
//单位
     struct PatNode        //HASH 表的结点
     {
          char pattern[64];      //模式串
          __int64 M[256];//位向量
          struct PatNode *next; //指针
     };
     struct PatNode *Hash[_SIZE_OF_TABLE]; //HASH 表
     struct Info       //保存匹配结果
     {
          int loc;     //匹配的位置
          char sentence[10]; //匹配的语句
     };
     struct Info m_info[5000];
     int infonum;     //统计共匹配多少次
private:
     int totallen;
void AddPatNode(WORD h, char *pat);//添加 HASH 结点
     int Search(struct PatNode *pNode, const char *text, int
tlen, int distance);//BPM 算法的实现
      WORD BKDRHash(char *str, int m);//BKDRHASH 散列
//算法
};
```

CHashBpm 类的主要实现代码如下：

```
int CHashBpm::AddPattern(char *pat)  //添加模式 pat
 {
     int len=0;
     WORD i=0,h=0;
     if (NULL == pat)
          return -1;
     len = strlen(pat);
     for (i=len-1;i>2;i-=2)
     {
     h = BKDRHash(pat+i-3, 4);//BKDR 算法求 HASH 值
          h &= 0x7fff;
           AddPatNode(h,pat);//添加 HASH 表的结点,结点保
//存待匹配的模式串 pat
     }
     return 0;
 }
   void CHashBpm::AddPatNode(WORD h, char *pat)// 添加
//HASH 表的结点
   {
     struct PatNode *pNode = NULL, *pTempNode=NULL;
     int i=0;
     __int64 offset=1;
     int patlen = strlen(pat);//模式 pat 的长度
     pNode = new struct PatNode;
     if (pNode)
     {
          strcpy(pNode->pattern, pat);
          memset(pNode->M, 0, sizeof(__int64)*256);//初始
//化位向量
          pNode->next = NULL;
          for (i=0;i<patlen;i++)
          {
               pNode->M[(BYTE)pat[i]] |= offset;//将模式串
//pat 向量化
               offset <<= 1;
          }
          if (NULL==Hash[h]->next)//将新建结点加入链表
               Hash[h]->next = pNode;
          else
          {
               pTempNode = Hash[h]->next;
               Hash[h]->next = pNode;
               pNode->next = pTempNode;
          }
     }
   }
   WORD CHashBpm::BKDRHash(char *str, int m)//BKDR 算
//法求 HASH 值
   {
          unsigned int seed = 131; //种子
          unsigned int hash = 0;
           int i=0;
          while (i<m)
          {
            hash = hash * seed + (*str++);
                 i++;
          }
          return (hash & 0xFFFF);
   }
   int CHashBpm::Match(const char *text, int distance)//多模
//式近似匹配
   {
```

```
    int textlen = strlen(text);   //文本的长度
    struct PatNode *pNode=NULL;
    int i=0, h=0, rlt=0;
    int maxlen=0,len=0;
    char tt[64];
    for (i=4;i<textlen;i+=2)
    {
        rlt = -1;
        maxlen = 1;
        h = BKDRHash((char *)text+i, 4);//求 HASH 值
        h &= 0x7fff;
        pNode = Hash[h]->next;
        while (pNode)//如存在入口地址,则调用 BPM 算法
//进行模式近似匹配
        {
            rlt = Search (pNode, text+i, strlen (pNode->
pattern)+distance, distance);
  //单模式近似匹配
            if (-1! =rlt)
            {
                len = strlen(pNode->pattern);
                if (maxlen < len)
                    maxlen = len;
                m_info[infonum].loc = totallen+i+rlt;
//保存匹配位置
            }
            pNode = pNode->next;
        }
        if (rlt>-1)
        {
            i+=rlt;
            strncpy(tt, text+i-distance, maxlen);
            tt[maxlen]=0;
            i+=(maxlen-distance);
            strcpy(m_info[infonum].sentence, tt);
            infonum++; //统计匹配次数
        }
    }
    totallen += textlen;
    return 0;
  }
  int CHashBpm::Search(PatNode *pNode, const char *text,
int tlen, int distance)
  //这是算法的核心,实现一个模式的近似匹配
  {
    int patlen = strlen(pNode->pattern); //模式串的长度
    int textlen = tlen; //文本的长度
    int score = patlen; //score 初始值为模式串的长度
    int lastMatchPos=-1;
    __int64 Pv, Mv;
    __int64 Xv, Xh, Mh, Ph;
    __int64 flag=(__int64)1<<(patlen-1);
    Pv = (__int64)-1;
    Mv = 0;
        for (int i=0;i<textlen;i++)//进行位向量运算,执行
//Myerstep 过程
    {
        int c=(BYTE)text[i];
        __int64 Eq = pNode->M[c];
        Xv = (Eq|Mv);
        Xh = (((Eq & Pv)+Pv)^Pv)|Eq;
        Ph = Mv|~(Xh|Pv);
        Mh = Pv & Xh;
        if (Ph & flag)
            score ++;//字符不同
        else if (Mh & flag)
            score --;//字符相同
        Ph<<=1;
        Mh<<=1;
        Pv = Mh|~(Xv|Ph);
        Mv = Ph & Xv;
        if (score <= distance) //误差在容许范围内
        {
        lastMatchPos = i+1-strlen(pNode->pattern)+2;
            return lastMatchPos; //返回匹配位置
        }
    }
    return lastMatchPos;
  }
  void CHashBpm::Release() //释放资源
  {
    PatNode *pNode=NULL;
    PatNode *pTmpNode=NULL;
    int i=0;
    for (i=0;i<_SIZE_OF_TABLE;i++) //释放链表
    {
        pNode = Hash[i]->next;
        while (pNode)
        {
            pTmpNode = pNode;
            pNode = pNode->next;
            delete pTmpNode;
        }
        Hash[i]->next = NULL;
    }
    for (i=0;i<infonum;i++) //清除匹配记录
    {
        m_info[i].loc=0;
        memset(m_info[i].sentence,0,10);
    }
    infonum=0;
  }
```

### 3.2 近似匹配的工程实现

用 VC++6.0 建立一个基于对话框的工程，工程采用多线程

技术，创建一个辅助线程 ThreadMatch () 用于模式的近似匹配，代码如下：

```
DWORD WINAPI ThreadMatch(LPVOID pVoid)
{
    int len=0;
    char buf[4096];
    int ret=0;
    pfin = fopen(filename,"rb");//打开待匹配的文件
    if (pfin ! = NULL)
    {
        len = fread(buf, 1, 4096, pfin); //读文件
        do
        {
            ret=m_bpm.Match(buf,distance); //进行匹配
        }while (len = fread(buf, 1, 4096, pfin));
        fclose(pfin);
    }
    SendMessage(hWnd, MSG_END, 0, 0);
    return 0;
}
```

工程的界面如图 3 所示，以在文本文件“Sports0029.txt”中近似匹配模式“运动科学”、“体育科研”、“认识论”为例，说明程序的用法。

(1) 如图 3 中标号 1 所示，点击“选择文件”按钮，选择待匹配的文件，文件可以是任意语种，上文以中文为例；

(2) 如图 3 中标号 2 所示，输入关键词“运动科学”、“体育科研”、“认识论”，容错字数=1，需要注意的是，容错字数不宜超过模式串长度的 40%，否则匹配结果的误差太大；

(3) 点击按钮“开始”，如图 3 中标号 3 所示，输出匹配结果，共找到多项近似匹配，最多只容许有一个错字，对比原始文件，结果正确。

图 3 工程应用示意图

## 4 结语

介绍了一种多模式的近似匹配算法——BKDR-BPM 算法的设计思路及简要实现方法，给出了该算法的工程实现，并完成了一个较为完整的近似匹配应用程序，可以支持任意语种。希望这篇文章能对从事模式匹配研究工作的编程爱好者们提供帮助。

## 参考文献

[1] S. Wu, U. Manber. A Fast Algorithm For Multi-Pattern Searching [J] . Technical Report TR-94-17, University of Arizona. 1994：1-11.

[2] http://www.byvoid.com/blog/string-hash-compare/.

[3] Myers G. A Fast Bit-Vector Algorithm for Approximate String Matching Based on Dynamic Programming [J] . Journal of the ACM Archive, 1999, 46 (3)：395-415.

（收稿日期：2012-01-17）

（上接第 16 页）

所示。

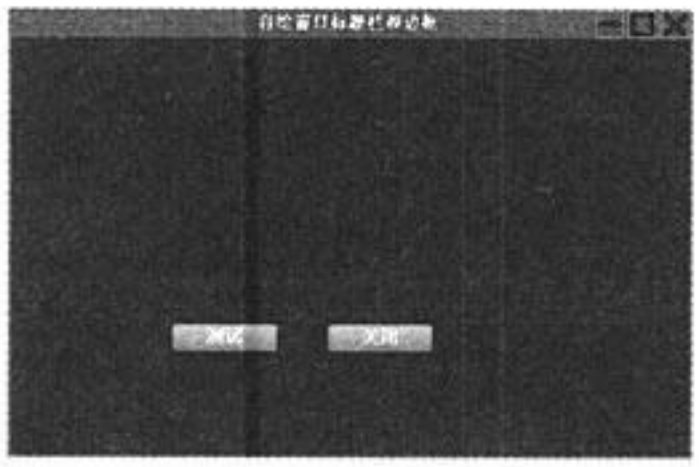


图 4 自定义重绘效果

## 6 结语

通过对 VC++编程知识和技术的分析、设计，可以了解 VC++应用程序尤其是 MFC 应用系统设计与开发的流程和解决方案；可以掌握 VC++编程技术和面向对象技术以及 bug 的调试技术和解决 bug 的能力，重点掌握了对话框应用程序界面重绘和控件重绘的知识和技术，尤其是重中之重的位图显示技术，同时也学会了如何设计并实现 VC++应用程序主界面的设计与美化。在设计界面过程中，渐渐地学会了如何设计漂亮、美观、友好的用户界面；最为重要的是通过这次设计与开发，使自己懂得如何在困难重重中一步一步细心地发现问题，解决问题。另外，提高了对编程认知与总结，不断加强编程基本功，不断总结经验，学习他人的优秀成果，并提高了独立思考和解决问题的能力。通过在软件设计开发中对用户界面的方向和用户心理的把握，来设计并开发出用户最为满意的软件程序。

（收稿日期：2012-12-09）

# 利用 PPT VBA 制作简单的演讲比赛系统

刘 烽

**摘 要**：对利用 PPT VBA 制作具有随机抽题和定时提醒功能的演讲比赛系统进行了探索，给出了具体的实现步骤和代码，该演讲比赛系统具有美观、易用、实用的特点。

**关键词**：PPT 文档；VBA 编程；演讲比赛；随机抽题

## 1 问题提出

笔者在参加几次演讲比赛活动中，不经意发现：每一次活动的比赛系统通常由系统开发人员使用高级编程语言（如 VB，C# 等）开发，然后打包生成可执行文件后交付举办方使用。这种方式对于举办方而言，往往会带来几个不便：（1）无法对系统做临时性、简单的修改（诸如更改系统背景、字体等）；（2）每次不同的演讲主题都需要重新修改源程序；（3）受开发人员编程能力和美工水平的限制，开发出的演讲比赛系统往往不够美观大方，演示效果不强。为解决这些实际中的矛盾，笔者探索使用 PPT VBA 开发一个简单的演讲比赛系统。相比用高级编程语言开发出的此类系统，用 PPT VBA 开发有以下几个优势：（1）简单易用，由于 PPT 是大家较为熟悉的软件，普通人就可对系统进行简单的更改，（2）系统界面便于修改，对不同的演讲主题，仅需修改一般文字性的内容，不必修改源程序。（3）由于 PPT 强大的美化和动画功能，使得系统的操作界面美观大方，有较好的演示效果。

## 2 实现步骤

### 2.1 系统需求及开发环境

简单的演讲比赛系统通常具有：（1）随机抽题；（2）倒计时；（3）定时提醒等几个基本功能。针对系统需求，笔者选定的开发环境是 PowerPoint 2007 + Windows XP。

### 2.2 实现步骤

（1）新建一个“空白 PPT 演示文稿”并将其另存为“启用宏的 PowerPoint 演示文稿（*.pptm）”，新建 3 页幻灯片，分别为标题页、随机抽题页和演讲题目页。标题页设置为演讲比赛的主题，比赛时间等，下面分别介绍随机抽题页和演讲题目页的设计。

（2）“随机抽题”页。“随机抽题”页为幻灯片第 2 页（图 1 所示），在幻灯片上插入 7 个形状，为形状“start”、“stop”、“open”插入“动作”，分别设置运行宏：Start、Result、OpenQus。7 个形状的属性设置如表 1 所示。

表 1 “随机抽题”页形状属性设置表

| 形状名称 | 是否可见 | 运行宏 | 说明 |
|---|---|---|---|
| M | 不可见 | - | 倒计时间的“分” |
| S | 不可见 | - | 倒计时间的“秒” |
| txtResult | 不可见 | - | 显示抽题结果 |
| txtSelected | 不可见 | - | 显示随机题号 |
| start | 可见 | Start | 开始抽题 |
| stop | 可见 | Result | 停止抽题 |
| open | 可见 | OpenQus | 打开选题 |

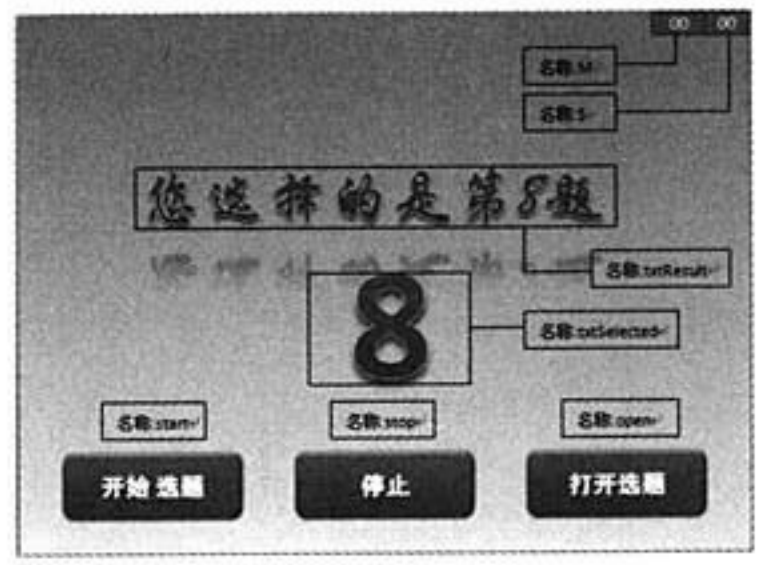



图 1 “随机抽题”页界面设计

点击 PowerPoint 2007“开发工具”选项卡中的“Visual Basic”工具栏，进行 VBA 开发环境，点击“插入”菜单，插入模块“模块 1”，在“模块 1”中编辑以下代码：

```
'模块 1
Option Explicit
Private Const n = 15 '总题目数
Dim a As Integer     '随机题号
Public bStop As Boolean
'宏 Start:开始抽题
Sub Start()
    ActivePresentation.Slides  (2).Shapes("txtSelected").Visible
= msoCTrue
    ActivePresentation.Slides (2).Shapes ("txtResult").Visible  =
```

```
msoFalse
    Randomize()
    Do
        a = Fix(Rnd * n + 1)
          ActivePresentation.Slides    (2).Shapes   ("txtSelected").
TextFrame.TextRange.Text = a
        DoEvents()
    Loop
End Sub
' 宏 OpenQus:打开对应的选题
Sub OpenQus() ActivePresentation.SlideShowWindow.View.
GotoSlide (Val (ActivePresentation.Slides (2).Shapes ("
txtSelected").TextFrame.TextRange.Text + 2))
End Sub
' 宏 Result:停止抽题
Sub Result()
    ActivePresentation.Slides  (2).Shapes("txtSelected").Visible
= msoFalse
    ActivePresentation.Slides (2).Shapes ("txtResult").Visible  =
msoCTrue
        ActivePresentation.Slides    (2).Shapes   ("txtResult").
TextFrame.TextRange.Text = "您选择的是第" & CStr(a) & "题"
    End
End Sub
```

在“随机抽题”页中插入一个“用户窗体”UserForm1（图 2 所示），用于设置选手的比赛时间，双击 UserForm1 后编辑以下代码：

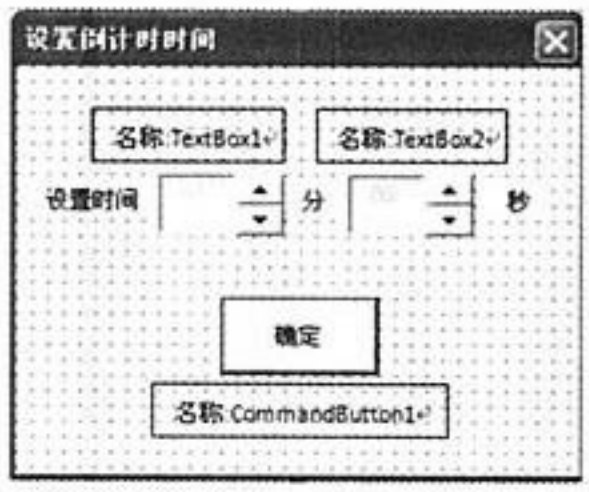



图 2 倒计时时间设置界面

```
Option Explicit
Private Sub CommandButton1_Click()
    If CInt  (Me.TextBox1.Text) = 0 And CInt (Me.TextBox2.
Text) = 0 Then
        MsgBox("请设置倒计时间！")
    Else
        Slide2.Shapes  ("M").TextFrame.TextRange.Text = Me.
TextBox1.Text
        Slide2.Shapes  ("S").TextFrame.TextRange.Text = Me.
TextBox2.Text
        Unload(Me)
    End If
End Sub
Private Sub SpinButton1_SpinDown()
    If CInt(Me.TextBox1.Text) > 0 Then
        Me.TextBox1.Text = Format(CInt(Me.TextBox1.Text) –
1, "00")
    Else
        Me.TextBox1.Text = Format(0, "00")
    End If
End Sub
Private Sub SpinButton1_SpinUp()
    If CInt(Me.TextBox1.Text) < 60 Then
        Me.TextBox1.Text = Format(CInt(Me.TextBox1.Text) +
1, "00")
    Else
        Me.TextBox1.Text = Format(60, "00")
    End If
End Sub
Private Sub SpinButton2_SpinDown()
    If CInt(Me.TextBox2.Text) > 0 Then
        Me.TextBox2.Text = Format(CInt(Me.TextBox2.Text) –
1, "00")
    Else
        Me.TextBox2.Text = Format(0, "00")
    End If
End Sub
Private Sub SpinButton2_SpinUp()
    If CInt(Me.TextBox2.Text) < 60 Then
        Me.TextBox2.Text = Format(CInt(Me.TextBox2.Text) +
1, "00")
    Else
        Me.TextBox2.Text = Format(60, "00")
    End If
     End Sub
```

在“工程-VBAProject”窗口，双击“Slide1”对象，编辑如下代码：

```
' 幻灯片播放时,这个宏首先并自动运行,主要完成初始化的工
' 作
Option Explicit
Sub OnSlideShowPageChange()
    ActivePresentation.Slides  (2).Shapes("txtSelected").Visible
= msoFalse
    ActivePresentation.Slides (2).Shapes ("txtResult").Visible  =
msoFalse
            If  ActivePresentation.SlideShowWindow.View.
CurrentShowPosition = 2 And _
            CInt  (Slide2.Shapes ("S").TextFrame.TextRange.
Text) = 0 And _
            CInt  (Slide2.Shapes ("S").TextFrame.TextRange.
Text) = 0 Then
        UserForm1.Show(0)
```

```
    End If
    Dim i As Integer
    For i = 3 To ActivePresentation.Slides.Count
        ActivePresentation.Slides (i).Shapes (1).TextFrame.
TextRange.Text = ""
    Next i
End Sub
```

(3) “演讲题目”页。“演讲题目”页为幻灯片第 3 页，完成倒计时和定时提醒功能。在幻灯片上插入 4 个形状，界面设置如图 3 所示，设置形状“开始计时”和“选题”的动作为运行宏“StartCount”和“GoToTitle”。在 VBA 代码编辑窗口新建模块“模块 2”，并编辑以下代码：

图 3 “演讲题目”页界面设置

```
Option Explicit
Declare Function SetTimer Lib "user32" (ByVal hwnd As
Long, ByVal nIDEvent As Long, ByVal uElapse As Long,
ByVal lpTimerFunc As Long) As Long
Declare Function KillTimer Lib "user32" (ByVal hwnd As Long,
ByVal nIDEvent As Long) As Long
Private Declare Function midiOutOpen Lib "winmm.dll"
(ByVal lphMidiOut As Long, ByVal uDeviceID As Long, ByVal
dwCallback As Long, ByVal dwInstance As Long, ByVal
dwFlags As Long) As Long
Private Declare Function midiOutClose Lib "winmm.dll"
(ByVal hMidiOut As Long) As Long
Private Declare Function midiOutShortMsg Lib "winmm.dll "
(ByVal hMidiOut As Long, ByVal dwMsg As Long) As Long
Private Declare Sub Sleep Lib "kernel32" (ByVal
dwMilliseconds As Long)
Public TimerID As Long
Public counts As Integer
Public isStart As Boolean
Sub TimerStart(ByVal t As Integer)
    counts = t
    TimerID = SetTimer(0, 0, 1000, AddressOf TimerProc)
    isStart = True
End Sub
Sub TimerStop()
    If isStart = False Then
        Exit Sub
    End If
    TimerID = KillTimer(0, TimerID)
    isStart = False
End Sub
Sub TimerProc (ByVal hwnd As Long, ByVal uMsg As Long,
ByVal idEvent As Long, ByVal byvaldwTime As Long)
    Dim M As Integer
    Dim S As Integer
    M = Int(counts / 60)
    S = counts Mod 60
            ActivePresentation.Slides (ActivePresentation.
SlideShowWindow.View.CurrentShowPosition).Shapes (1).
TextFrame.TextRange.Text = "剩余时间:" & Format(M, "00:")
& Format(S, "00")
    counts = counts - 1
    If (counts < -1) Then
        TimerID = KillTimer(0, TimerID)
            ActivePresentation.Slides (ActivePresentation.
SlideShowWindow.View.CurrentShowPosition).Shapes (1).
TextFrame.TextRange.Text = "时间到！"
        Sleep(500)
        Bell(2)
    End If
End Sub
Sub StartCount()
    Dim TotalTime As Integer
    TimerStop()
        TotalTime = CInt (Slide2.Shapes("M").TextFrame.
TextRange.Text) * 60 + CInt (Slide2.Shapes("S").TextFrame.
TextRange.Text)
    TimerStart(TotalTime)
End Sub
' 模拟打铃 n 声
Private Sub Bell(ByVal n As Integer)
    Dim Midi As Long, M_HMidi As Long
    Dim i As Integer
    Dim t1 As Long
    Midi = midiOutOpen(M_HMidi, -1, 0, 0, 0)
    For i = 1 To n
        Midi = midiOutShortMsg(M_HMidi, &H7F2890)
        t1 = Timer
        While Timer - t1 < 1
            DoEvents()
        End While
    Next i
    Midi = midiOutClose(M_HMidi)
End Sub
' 宏 GoToTitle:返回抽题页
Sub GoToTitle()
    ActivePresentation.SlideShowWindow.View.GotoSlide (2,
msoCTrue)
    ActivePresentation.Slides (2).Shapes ("txtResult").Visible =
```

(下转第 36 页)

# 基于JavaEE技术的“医院检验分析系统”设计与实现

陈 震

**摘 要**：说明了“医院检验分析系统”的必要性，从该项目需要实现的功能入手，介绍了系统的建模设计过程，结合具体的系统模块详细分析了核心和难点的代码实现细节，总结了该项目的特点和需要完善的功能。

**关键词**：细菌耐药分析；JavaEE技术；OOAD方法；持久化对象；分层开发

## 1 项目的设计背景和实用价值

现在很多大中型医院都实施了HIS系统（医院信息管理系统），并且在检验科实施了LIS系统（医学实验室信息系统）与HIS系统配合，来完善医院的信息化建设，减少人为操作的失误，提高工作效率，从而让医院日常工作运转高效合理。然而每个医院都有自身的特点，而LIS等系统又往往是产品化的而无法满足这些重要的个性需求。比如，在很多医院“细菌的耐药统计分析”就是检验人员的一个重要的日常性工作，这项工作要对病人的标本按照药物、酶、医院科室、日期等多种复杂逻辑进行统计，并在全院定期汇总交流，辅助医生临床诊断，并形成归档文件长期保存供日常查阅。为此，医院检验科每个月都要安排多名检验师用10天左右的时间人工加班录入大量标本数据到Execl表格，并在该Execl表格中完成部分数据的统计，然后再人工计算剩余不能统计的数据，最后，编写汇总报告上交全院交流存档。这样的工作每个月都要重复地进行，占用了检验师大量的时间，浪费了医院大量的人力财力，并且，如果要汇总多个月份、季度、年份的数据，这种重复劳动还得再次进行，效率非常低下。

多个医院检验科强烈地反映了这种需求以后，经过认真分析调研，决定开发一套“医院检验分析系统”来弥补现有LIS系统的不足。该系统的投入使用进一步提升了医院检验工作的效率，促进了医院各部门工作更加有效地开展，并且为医院节约了大量的人力物力开销。

## 2 技术选型与原因

从时间、成本和规范化等因素考虑，采用了JavaEE技术来开发整套“医院检验分析系统”。JavaEE技术经过十多年的发展，已经非常成熟，在医院、金融、电信等大中型企业级应用开发中占有绝对的地位，同时由于JavaEE对面向对象编程的完美支持以及其开放源代码的特点，全世界拥有众多的支持者、社区团体和公司，多年来涌现了大量优秀的第三方开源框架，比如spring、hibernate、struts、mybatis、ofbiz、jbpm等，这些开源框架不仅免费，而且其代码质量和设计思想之先进更是超越了很多商业软件系统，多年来一直引领着整个软件行业的发展方向。

设计这套“医院检验分析系统”时，并没有采用重量级JavaEE设计模型EJB3.0，而是选用了轻量级JavaEE设计模型SSH2（Stuts2 + Spring + Hibernate），这样的选择主要是由于SSH2模型相比EJB3.0模型已经足够成熟，并且非常规范，而且在开发周期和开发成本上SSH2还有较大的优势，而它们的差别：主要是EJB3.0模型在分布式事务和EJB容器等技术上的优势，这些优势对于大多数软件系统却很少触及。使用JavaEE技术架构来组织大型软件系统的开发，不仅可以沿用其在软件行业成熟的项目管理方法、代码开发的规范、文档撰写的标准，而且可以站在众多成熟的中间件技术的巨人肩膀之上，而不被“百层大楼从零起”的传统开发方式所困，为此，在该项目中引入了许多成熟高效的中间件技术，比如，数据访问，采用了成熟的ORMapping框架（对象关系映射）Hibernate；这些中间件的融入，使系统更加的成熟和模块化，并且大大地提高了开发效率。不仅如此，该系统还有着科学而严谨的分层架构，整个系统严格遵循MVC（模型–视图–控制器）设计模式，同时，为了清晰地区分业务逻辑和兼顾代码的可读性，系统的所有组件（对象模型）都由Spring容器（对象工厂）统一管理，这样的设计从全局角度保证了系统代码的长期可维护性，是业内公认最为先进的软件设计方法。数据库采用了Mysql5.1，mysql经过多年的发展，已经发展为一个成熟的关系数据库系统，它在稳定性、效率、高可用性等多个方面具有非常优秀的表现，比如,它的单位时间数据库事务完成率已经超越很多商业数据库产品；它的master-slave replication高可用技术（主从数据库实时复制技术）对于构建分布式的应用系统很有意义。由于在轻量级SSH2开发中，spring充当了JavaEE容器的作用，因此，没有必要采用昂贵的Weblogic、WebSphere等EJB容器，采用的Tomcat JSP容器就足以满足业

务需求。

## 3 需求分析（OOAD 领域建模）

在开发“医院检验分析系统”之前，首先要熟悉医院的检验业务以及相关的重要概念。

医院检验科每隔一段时间就会从临床分离大量的病原菌株标本，并对纳入医院目标性监测的多重耐药菌进行统计分析。这些分析包括菌株分布、标本分布、科室分布、多重耐药菌株的耐药情况分析等，同时，还要对这些用于检测的菌株标本进行培养，根据其敏感性撰写赖药检测报告。

在“细菌赖药分析”的过程中，存在多个供菌株标本使用的、同时不变的基础数据，它们是：细菌（Bacterium）、细菌类型（BacteriumType）、菌株标本类型（SpecimenType）、抗生素或药物（Drug）、酶（Enzyme）、医院科室（Department）。这些基础数据和菌株标本之间以及基础数据内部都存在着错综复杂的关联关系。这种关系正是我们进行系统建模的关键所在。“细菌赖药分析”的整个过程始终以菌株标本数为计量单位，来统计某种标本分布的绝对数目或者相对百分比，每 1 个标本都会与 1 种细菌、1 种标本类型、多种抗生素或药物、多种酶以及 1 个医院科室产生关联，同时每个标本关联的多种抗生素和多种酶必须由标本关联的细菌类型来决定。在初步了解了以上需求后，运用 OOAD（面向对象分析与设计）方法对其进行了需求调研。采用 OOAD 的方法进行需求分析，不仅可以使需求调研科学高效（面向对象思想符合人脑思维，减少理解的奇异，降低沟通成本，易于形成甲乙双方共同认可的开发文档），而且能和后期的软件开发进行有机整合（JavaEE 规范本身就是面向对象软件开发的集大成者）。在分析过程中大量运用了 UML（OOAD 使用的主要工具语言）的时序图、对象图等和使用者进行交流，并形成了最终需求调研文档，如图 1 所示。

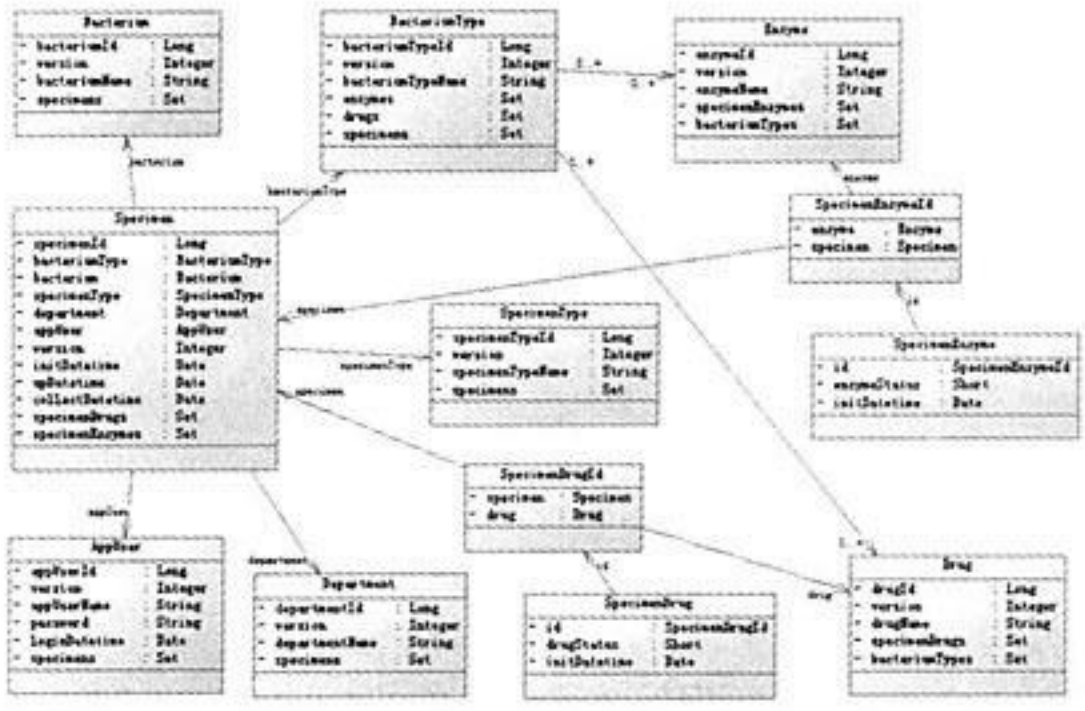



图 1 UML 实体对象图

最终，我们在模型层（Model）把“细菌赖药分析”需要的数据归纳为 12 个实体对象（POJO），它们分别是：AppUser（系统用户对象）、Bacterium（细菌对象）、BacteriumType（细菌类型对象）、Department（医院科室对象）、Drug（抗生素或药物对象）、Enzyme（酶对象）、Specimen（菌株标本对象）、SpecimenDrug（标本药物对象）、SpecimenDrugId（标本药物标识对象）、SpecimenEnzyme（标本酶对象）、SpecimenEnzymeId（标本酶标识对象）、SpecimenType（标本类型对象）。

这些实体对象之间存在以下复杂关系：AppUser、Bacterium、BacteriumType、Department、SpecimenType 4 个对象和 Specimen 对象之间是 1 对多双向关联关系；BacteriumType 对象和 Drug、Enzyme 2 个对象之间是多对多单向关联关系；Specimen 和 Drug 之间本来可以用多对多关系实现，但由于要保存多对多关系的状态属性，因此，在它们之间加入了 SpecimenDrug 对象，并且 SpecimenDrug 对象的主键由复合主键对象 SpecimenDrugId 来标识，在 SpecimenDrugId 复合主键对象中存在与 Specimen、Drug 对象的多对 1 双向关联，因此，SpecimenDrug 对象也同样产生了与 Specimen、Drug 对象的多对 1 双向关联（由于 SpecimenDrugId 是 SpecimenDrug 的主键）；Specimen 和 Enzyme 之间的关系与上相似。

经过这样的设计以后，“细菌赖药分析”所需要的报表数据就可以通过这 12 个实体对象之间的关联和运算很好地展现，同时由于面向对象建模是对业务系统最为真实而自然的描述，因此其稳定性也是可以保证的，即使业务模型有变化，也可以控制在每个对象实体内部，而不会对业务系统的其他模块造成较大影响。

## 4 数据库分析

在采用 OOAD 方法对“细菌赖药分析”的业务需求进行合理的分析以后，生成数据库物理模型（数据库表结构）就显得非常轻松了，但由于对象模型和物理模型（关系模型）之间仍然存在着较大区别，必须在对象模型和物理模型之间进行数据的转换。

这种转换必须遵循一定的规范才能进行，比如 1 个对象实体通常情况下可以直接和数据库的 1 张表对应；2 个对象实体间的 1 对多或者多对 1 关系可以转换为表之间的主外键参照关系；2 个对象之间的多对多关系（没有附加属性的情况）在转换为表结构时，会创建 3 张表，分别是 2 个对象对应的 2 张实体表以及 1 张中间关联表，这张中间表只能有 2 个属性（每个属性作为外键参照对应实体表，同时这两个属性组成复合主键唯一标识本表记录）；如果 2 个对象之间是存在附加属性的多对多关系（比如，需要记录关联的时间、状态等属性），这种情况下，建议把 2 个多对多的对象变换为 3 个对象来建模，加入 1 个中间关联对象，就这个中间对象而言形成了 2 个多对 1 的关系，在这个中间对象中就可以加入多个附加属性了，而且，这个中间对象的标识建议采用一个复合主键对象来实现，

而对应的数据库仍然只有 3 张表。

鉴于以上转换规范，把需求分析阶段得到的对象模型转换为了以下 12 张数据库表，如表 1 所示。

**表 1　实体对象和数据库表的转换图**

| 对象模型 | 数据库表 |
|---|---|
| AppUser | app_user |
| Bacterium | bacterium |
| BacteriumType | bacterium_type |
| Department | department |
| Drug | drug |
| Enzyme | enzyme |
| Specimen | specimen |
| SpecimenDrug | specimen_drug |
| SpecimenDrugId | |
| SpecimenEnzyme | specimen_enzyme |
| SpecimenEnzymeId | |
| SpecimenType | specimen_type |
| | bateriumtype_drug |
| | bateriumtype_enzyme |

从表 1 可以看出，对象和数据库表并非完全对应，因此，必须遵循以上规范才能把对象模型转换为如图 2 所示的合理关系模型：

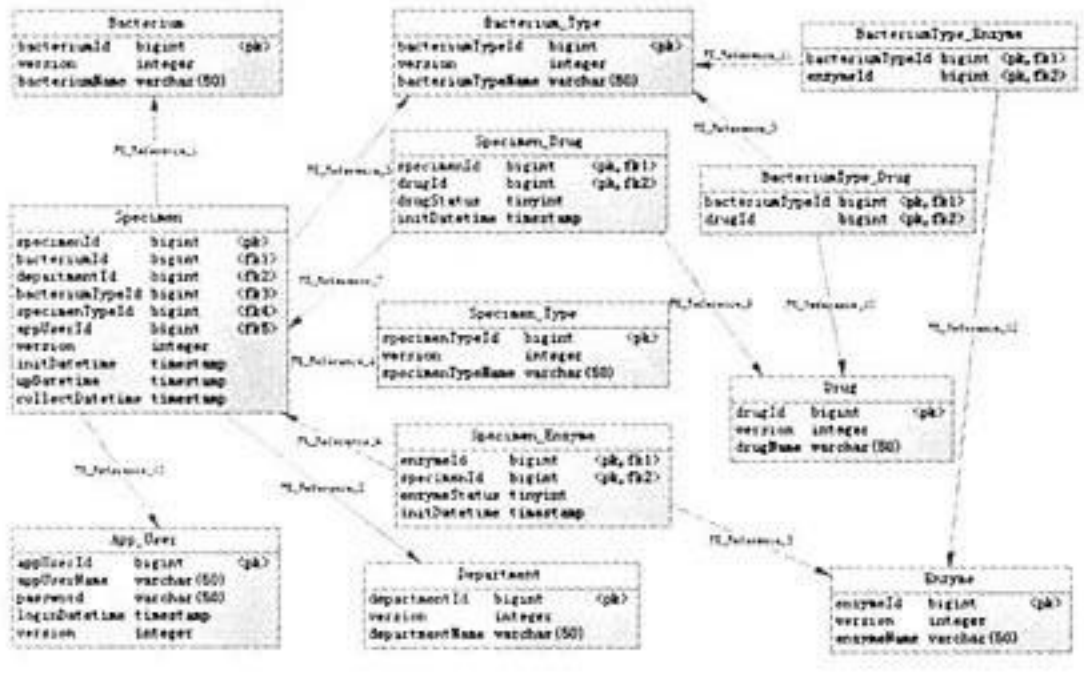



**图 2　数据库表结构图**

## 5　软件架构

整个系统分为 dao (数据层)、service (业务层)、action (表现层) 3 层架构，并且将业务数据封装在 12 个模型对象 (只有数据，没有业务逻辑) 中并在 3 层之间流转，同时把常用的操作定义为工具类封装在 util 包中供其他类调用，而且把 dao、service、action 层都要使用的增、删、改、查等基础代码采用 JavaEE1.5 的泛型机制和面向对象的接口机制进行封装和解耦，每层和每个模块之间都面向接口编程，这样的设计极大地提升了代码的健壮性、重用性和可读性，保证了项目在扩展时，仍然具有良好的结构。在 dao 层和数据库之间采用了 hibernate 框架（ORM，对象关系映射框架）与底层数据库隔离，这样的系统不仅可以在不同的操作系统上运行，而且可以在不同的数据库平台上部署，构建了一个十分优秀的弹性扩展平台。整个业务系统的所有组件（对象、接口、配置文件等）都统一由 Spring（依赖注入容器，自动生成和管理项目相关的组件，不用手工生成）容器管理，同时，把数据库事务、用户登录、日志等具有横切性质的业务逻辑也由 Spring 容器的 Aop (面向切面编程) 机制实现，从而使代码分工更加明确，系统的可维护性进一步提升。表现层采用 Struts2 的 MVC 模式实现，而 jsp 页面不会包含任何业务逻辑，只用于数据的录入和展现，复杂的业务逻辑也不在 Struts2 的 action 中实现，而是封装在 service 层的业务类中实现，action 层通过调用 service 层接口 (interface) 的方式访问这些复杂逻辑，然后，显示给最终用户，系统分层如图 3 所示。

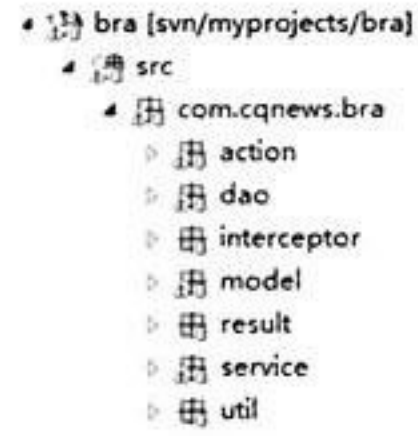



**图 3　系统分层图**

## 6　核心及难点代码解析

### 6.1　系统重要配置文件

在 JavaEE 开发的项目中，推荐使用配置文件或者注解的方式把一些容易变化的因数从程序代码中分离出来，这样的设计让软件系统各组件之间耦合度降低，系统扩展性增强。在轻量级 JavaEE 框架 SSH2 中有以下一些重要的配置文件需要特别注意：

#### 6.1.1 Web.xml

每个 JavaEE Web 项目都有 web.xml 文件，这是由 JavaEE 规范定义的，该文件中可以配置 filter（过滤器）、servlet、listener（监听器），并且 spring、struts 等很多著名框架也是以这个文件作为入口点进行配置的。

Web.xml 位于每个 Web 项目根目录下的 WEB-INF 目录中。

文件的头部必须包含以下命名空间：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/
javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
```

以下参数定义 log4j 日志信息：

```
<context-param>
    <param-name>webAppRootKey</param-name>
    <param-value>bra</param-value>
</context-param>
<context-param>
    <param-name>log4jConfigLocation</param-name>
<param -value >classpath:conf/log4j.properties </param -
value>
</context-param>
<context-param>
    <param-name>log4jRefreshInterval</param-name>
    <param-value>60000</param-value>
</context-param>
```

以下配置是整个文件中最为重要的部分，ContextLoaderListener 监听器的作用就是在启动 Web 容器时，自动加载 ApplicationContext（代表 spring 的上下文信息的对象）的配置信息，并且还定义了 Spring 框架的入口参数和相应要读取的配置文件，是启动 Spring 框架的入口点，由于自己的程序主要基于 Spring 框架开发，因此，也由此而启动：

```
<listener>
    <listener-class>
            org.springframework.web.context.
ContextLoaderListener
    </listener-class>
</listener>
<context-param>
    <param-name>contextConfigLocation</param-name>
<param -value >classpath:conf/app -context.xml </param -
value>
</context-param>
```

配置 jstl 标签，jstl 是一种在表现层 jsp 页面中输出后台数据的常用标签，使用 jstl 可以避免在 jsp 页面中写大量凌乱的逻辑代码：

```
<jsp-config>
    <taglib>
<taglib-uri>http://www.cqnews.net/jstl/core</taglib-uri>
    <taglib-location>/WEB-INF/c.tld</taglib-location>
    </taglib>
    <taglib>
<taglib-uri>http://www.cqnews.net/jstl/fmt</taglib-uri>
    <taglib-location>/WEB-INF/fmt.tld</taglib-location>
    </taglib>
    <taglib>
    <taglib-uri>http://www.cqnews.net/jstl/fn</taglib-uri>
    <taglib-location>/WEB-INF/fn.tld</taglib-location>
    </taglib>
</jsp-config>
```

### 6.1.2 app-context.xml

app-context.xml 是 SSH2 轻量级 JavaEE 框架最重要的配置文件。spring 容器就是通过 ContextLoaderListener 监听器在 web 容器（Tomcat 等）启动时读取 app-context.xml 配置文件完成系统组件的初始化：

```
<?xml version="1.0" encoding="UTF-8"?>
   <beans xmlns="http://www.springframework.org/schema/
beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:context="http://www.springframework.org/
schema/context"
     xsi:schemaLocation = "http://www.springframework.org/
schema/beans
                http://www.springframework.org/schema/
beans/spring-beans.xsd
                    http://www.springframework.org/schema/
contexthttp://www.springframework.org/schema/context/
spring-context-2.5.xsd">
    <import resource="app-resources.xml" />
    <import resource="app-dao.xml" />
    <import resource="app-service.xml" />
</beans>
```

该文件结构清晰，采用 import 的方式按系统的分层结构又包含了 3 个配置文件：app-resources.xml 文件主要配置数据库连接等相关的信息；app-dao.xml 文件主要配置数据层代码的实现类；app-service.xml 文件主要配置业务层代码的实现类，以及业务层的数据库事务特性。

### 6.1.3 app-resources.xml

以下配置读取 jdbc.properties 属性文件获取数据库地址、用户名、密码等信息，properties 文件是项目开发中经常使用的文件，该文件结构简单，易于维护，通常用于配置变量信息：

```
<bean id="propertyConfigurer"
          class = "org.springframework.beans.factory.config.
PropertyPlaceholderConfigurer">
    <property name="locations">
        <list>
        <value>classpath:conf/jdbc.properties</value>
        </list>
    </property>
</bean>
```

在 id 为“dataSource”的 bean 中，包含了大量用“${}”代表的变量，这些变量就是从 jdbc.properties 属性文件中读取的，其中使用了 c3p0 数据库连接池来管理数据库连接，名为“dataSource”的 bean 也会在 spring 容器中被其他 bean 作为参数使用：

```
<bean id="dataSource"
   class="com.mchange.v2.c3p0.ComboPooledDataSource">
    <property name="driverClass">
        <value>${jdbc.driverClassName}</value>
    </property>
```

```
<property name="jdbcUrl">
    <value>${jdbc.url}</value>
</property>
<property name="user">
    <value>${jdbc.username}</value>
</property>
<property name="password">
    <value>${jdbc.password}</value>
</property>
<property name="initialPoolSize">
    <value>${c3p0.initialPoolSize}</value>
</property>
<property name="minPoolSize">
    <value>${c3p0.minPoolSize}</value>
</property>
<property name="maxPoolSize">
    <value>${c3p0.maxPoolSize}</value>
</property>
<property name="maxIdleTime">
    <value>${c3p0.maxIdleTime}</value>
</property>
<property name="acquireIncrement">
    <value>${c3p0.acquireIncrement}</value>
</property>
<property name="maxStatements">
    <value>${c3p0.maxStatements}</value>
</property>
<property name="idleConnectionTestPeriod">
    <value>${c3p0.idleConnectionTestPeriod}</value>
</property>
<property name="acquireRetryDelay">
    <value>${c3p0.acquireRetryDelay}</value>
</property>
<property name="acquireRetryAttempts">
    <value>${c3p0.acquireRetryAttempts}</value>
</property>
<property name="breakAfterAcquireFailure">
    <value>${c3p0.breakAfterAcquireFailure}</value>
</property>
<property name="testConnectionOnCheckout">
    <value>${c3p0.testConnectionOnCheckout}</value>
    </property>
    <property name="numHelperThreads">
        <value>${c3p0.numHelperThreads}</value>
    </property>
</bean>
```

id 为“sessionFactory”的 bean 中就引用了“dataSource”，用于构造一个代表数据库的工厂对象。由于项目使用了 hibernate 框架来映射关系数据库，因此，这个 bean 还必须加载 hibernate 相关的信息：“classpath*:com/cqnews/bra/model/*.hbm.xml”就代表要加载的实体对象的 hbm 配置文件；“hibernate.dialect”代表底层的数据库类型；“hibernate.show_sql”代表在系统的控制台实时显示 sql；“hibernate.hbm2ddl.auto”代表在系统启动时是否创建或者更新数据库表结构；还可以在该文件中配置缓存等信息：

```
<bean id="sessionFactory"
                class = "org.springframework.orm.hibernate3.
LocalSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="mappingLocations">
        <list>
            <value >classpath*:com/cqnews/bra/model/*.
hbm.xml</value>
        </list>
    </property>
    <property name="hibernateProperties">
        <props>
    <prop key="connection.useUnicode">true</prop>
<prop key="connection.characterEncoding">utf-8</prop>
            <prop key="hibernate.dialect">
                ${hibernate.dialect}
            </prop>
         <prop key="hibernate.show_sql">true</prop>
        <prop key="hibernate.jdbc.batch_size">20</prop>
        <prop key="hibernate.jdbc.fetch_size">20</prop>
            <prop key="hibernate.cache.provider_class">
                org.hibernate.cache.EhCacheProvider
            </prop>
                        <prop  key = "net.sf.ehcache.
configurationResourceName">
                conf/ehcache.xml
            </prop>
                        <prop  key = "hibernate.cache.
use_second_level_cache">
                true
            </prop>
    <prop key="hibernate.hbm2ddl.auto">update</prop>
        </props>
    </property>
</bean>
```

6.1.4 Jdbc.properties

```
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc\:mysql\://127.0.0.1\:3306/bra?useUnicode
true&characterEncoding
utf-8
hibernate.dialect=org.hibernate.dialect.MySQLInnoDBDialect
jdbc.username=bra
jdbc.password=cqnews
c3p0.acquireIncrement=5
c3p0.initialPoolSize=3
c3p0.idleConnectionTestPeriod=60
```

```
c3p0.minPoolSize=2
c3p0.maxPoolSize=50
c3p0.maxStatements=0
c3p0.numHelperThreads=10
c3p0.maxIdleTime=60
c3p0.acquireRetryDelay=1000
c3p0.acquireRetryAttempts=30
c3p0.breakAfterAcquireFailure=true
c3p0.testConnectionOnCheckout=false
```

6.1.5 app-dao.xml

该文件主要用于配置数据层代码的实现类，由于系统不同组件之间是面向接口编程的，因此可以通过向相同接口配置不同实现类的方式实现完成不同功能的组件的替换。

文件首先申明了 id 为："genericDao" 的 bean，并为其配置了 GenericDaoImpl 实现类，然后又为 id 为 "baseDao" 的 bean 配置了 BaseDaoImpl 实现类，并继承 "genericDao"，后面的 bean 也都有相应的实现类，并继承 "baseDao"。

```
<bean id="genericDao" abstract="true"
      class="com.cqnews.core.dao.impl.GenericDaoImpl">
      <property name="jdbcTemplate" ref="jdbcTemplate" />
<property name="sessionFactory" ref="sessionFactory" />
</bean>
<bean id="baseDao" abstract="true"
          class = "com.cqnews.core.dao.impl.BaseDaoImpl"
parent="genericDao" />
<bean id="bacteriumDao"
         class = "com.cqnews.bra.dao.impl.BacteriumDaoImpl"
parent="baseDao" />
<bean id="bacteriumTypeDao"
                          class = "com.cqnews.bra.dao.impl.
BacteriumTypeDaoImpl" parent="baseDao" />
<bean id="departmentDao"
     class = "com.cqnews.bra.dao.impl.DepartmentDaoImpl"
parent="baseDao" />
<bean  id = "drugDao"  class = "com.cqnews.bra.dao.impl.
DrugDaoImpl"
      parent="baseDao" />
<bean  id = "enzymeDao"  class = "com.cqnews.bra.dao.impl.
EnzymeDaoImpl"
      parent="baseDao" />
<bean id="specimenDao"
         class = "com.cqnews.bra.dao.impl.SpecimenDaoImpl"
parent="baseDao" />
<bean id="specimenDrugDao"
                          class = "com.cqnews.bra.dao.impl.
SpecimenDrugDaoImpl" parent="baseDao" />
<bean id="specimenEnzymeDao"
                          class = "com.cqnews.bra.dao.impl.
SpecimenEnzymeDaoImpl"
      parent="baseDao" />
<bean id="specimenTypeDao"
                          class = "com.cqnews.bra.dao.impl.
SpecimenTypeDaoImpl" parent="baseDao" />
<bean  id = "appUserDao"  class = "com.cqnews.bra.dao.impl.
AppUserDaoImpl"
      parent="baseDao" />
```

6.1.6 app-service.xml

该文件主要用于配置业务层的实现类以及数据库事务特性。由于系统的核心业务逻辑总在业务逻辑层实现，并且业务逻辑往往是一个整体不能分割，因此，在该层配置数据库事务特性是非常必要的。

以下配置引入了数据库事务特性，比如设置了 "com.cqnews.bra.service" 包下所有类的方法（除去名字以 get、is、find 开头的方法）具有事务特性：

```
<aop:aspectj-autoproxy />
<context:annotation-config />
<context:component-scan base-package="com.cqnews.bra.
service" />
<tx:annotation-driven transaction-manager="txManager" />
<bean id="txManager"
                class="org.springframework.orm.hibernate3.
HibernateTransactionManager">
<property name="sessionFactory" ref="sessionFactory" />
</bean>
<aop:config>
     <aop:pointcut id="servicePointCut1"
expression="execution(* com.cqnews.bra.service..*(..))" />
     <aop:pointcut id="servicePointCut2"
          expression = "execution (* com.cqnews.core.
service..*(..))" />
     <aop:advisor advice-ref="txAdvice"
          pointcut-ref="servicePointCut1" />
     <aop:advisor advice-ref="txAdvice"
          pointcut-ref="servicePointCut2" />
</aop:config>
<tx:advice id="txAdvice" transaction-manager="txManager">
     <tx:attributes>
          <tx:method name="get*" read-only="false" />
          <tx:method name="is*" read-only="true" />
          <tx:method name="find*" read-only="true" />
          <tx:method name="*" />
     </tx:attributes>
</tx:advice>
```

接下来，为 id 为 "bacteriumService" 的 bean 配置了 "BacteriumServiceImpl" 实现类，并且通过构造方法为其注入 "bacteriumDao" 的 bean（该 bean 已在 app-dao.xml 中配置）：

```
<bean id="bacteriumService"
                          class = "com.cqnews.bra.service.impl.
BacteriumServiceImpl">
```

```
    <constructor-arg index="0" ref="bacteriumDao" />
</bean>
```

### 6.1.7 Struts.xml

每个 SSH2 项目中的 WEB-INF/classes 目录下都有 struts.xml 文件，该文件是 struts2 框架的主要配置文件。struts2 框架是 JavaEE 开发中使用频率非常高的 MVC 框架，它主要作为 MVC 模式中的控制器接收用户提交的请求，并调用业务层的逻辑进行处理，最后把业务层处理的结果返回给前端 jsp、freemaker (一种常用的表现层模板技术) 等显示。因此，该文件主要配置与前端显示相关的一些组件。

以下配置主要定义用户访问 url 地址的后缀名为 “*.do”，以及启用开发者模式，输出调试信息：

```
<constant name="struts.action.extension" value="do" />
<constant name="struts.devMode" value="true" />
```

以下定义了 struts2 作为控制器工作的主要信息，比如 url 为" /usercenter" 目录采用拦截器进行权限控制；报错后，跳转到的页面等：

```
<package name="usercenter" extends="struts-default"
    namespace="/usercenter">
    <interceptors>
        <interceptor name="UserCenterInterceptor"
                class="com.cqnews.bra.interceptor.
UserCenterInterceptor" />
   <interceptor-stack name="UserCenterInterceptorStack">
 <interceptor-ref name="defaultStack"></interceptor-ref>
                <interceptor-ref name="
UserCenterInterceptor"></interceptor-ref>
        </interceptor-stack>
    </interceptors>
                <default-interceptor-ref name="
UserCenterInterceptorStack" />
    <global-results>
    <result name="myerror">/WEB-INF/error.jsp</result>
        <result name="login">/WEB-INF/usercenter/login.
jsp</result>
    <result name="input">/WEB-INF/error.jsp</result>
    </global-results>
    <global-exception-mappings>
     <exception-mapping exception="java.lang.Exception"
            result="myerror" />
    </global-exception-mappings>
    <action name="login_*" class="com.cqnews.bra.action.
LoginAction"
        method="{1}">
<result name="success" type="redirect">index.do</result>
        <result name="input">/WEB-INF/usercenter/login.
jsp</result>
        <interceptor-ref name="defaultStack" />
    </action>
<action name="index"
    class="com.cqnews.bra.action.IndexAction">
    <result>/WEB-INF/usercenter/index.jsp</result>
</action>
<action name="admin_top"
    class="com.cqnews.bra.action.TopAction">
<result>/WEB-INF/usercenter/admin_top.jsp</result>
</action>
<action  name = "left"  class = "com.cqnews.bra.action.
LeftAction">
        <result>/WEB-INF/usercenter/left.jsp</result>
    </action>
    <action name="right">
        <result>/WEB-INF/usercenter/right.jsp</result>
    </action>
    <action name="appuser_*"
        class  =  "com.cqnews.bra.action.AppUserAction"
method="{1}">
        <result name="toUpdate">
            /WEB-INF/usercenter/appuser_update.jsp
        </result>
        <result name="update" type="redirect">
            appuser_toUpdate.do
        </result>
    </action>
    <action name="bacteriumType_*"
                class = "com.cqnews.bra.action.
BacteriumTypeAction" method="{1}">
    </action>
    <action name="specimen_*"
            class = "com.cqnews.bra.action.SpecimenAction"
method="{1}">
        <result name="toCreate">
            /WEB-INF/usercenter/specimen_create.jsp
        </result>
        <result name="create" type="redirect">
            specimen_toCreate.do
        </result>
    </action>
    <action name="enzyme_*"
            class = "com.cqnews.bra.action.EnzymeAction"
method="{1}">
        <result name="list">
    /WEB-INF/usercenter/enzyme_drug_sensitive_list.jsp
        </result>
        <result name="toCreate">
            /WEB-INF/usercenter/specimen_create.jsp
        </result>
    </action>
    <action name="bacterium_*"
            class="com.cqnews.bra.action.BacteriumAction"
method="{1}">
```

```
        <result name="list">
    /WEB-INF/usercenter/bacterium_drug_sensitive_list.jsp
        </result>
        <result name="toCreate">
            /WEB-INF/usercenter/specimen_create.jsp
        </result>
    </action>
</package>
```

### 6.2 “菌株标本录入”模块的难点及核心代码

从需求分析可知，“菌株标本录入”模块存在几个难点：

难点一：标本录入界面的设计。用户选择不同细菌类型时，药品和酶的种类在变化，界面必须根据细菌类型动态变化，并且1个细菌类型对应多种药品和多种酶，每种药品又具有“敏感”、“耐药”、“中介”3种状态，每种酶也具有“阳性”、“阴性”两种状态，如图4所示。

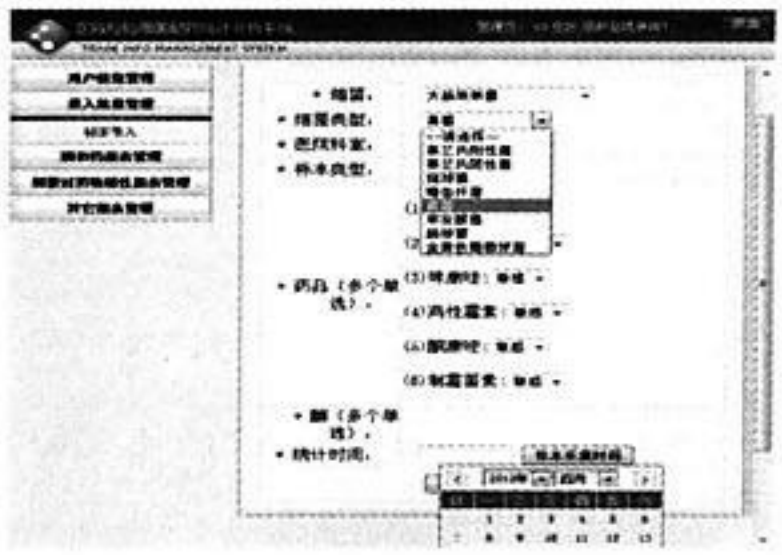


图4 “菌株标本录入”的界面图

难点二：标本保存业务逻辑的设计。由于一个标本的保存牵涉到需求分析阶段12种实体对象的复杂逻辑关系，因此，标本的保存也是比较复杂的。

#### 6.2.1 标本录入界面

由于药物和酶会随用户选择的细菌类型动态地变化，因此，在jsp页面中采用Ajax技术向服务器程序异步获取数据来更新用户前端界面是不二的选择。

首先，在specimen_create.jsp页面<body>中相应位置加入id为“drug”和id为“enzyme”的html标签，这两个标签主要用于js代码动态生成多个药物和酶的下拉单选框时的占位标记：

```
<span id="drug"></span>
<span id="enzyme"></span>
```

然后，在specimen_create.jsp页面的<head>内部加入两个javascript函数bacteriumType_clear () 和bacteriumType_change () ，bacteriumType_clear () 函数完成占位标记的初始化清理工作，bacteriumType_change () 函数通过响应细菌类型下拉列表的onChange () 事件完成药品和酶的动态加载：

```
function bacteriumType_clear() {
  var drug = document.getElementById("drug");
  var sLength = drug.length;
  for(i=0;i<sLength;i++) {
   drug.options[i] = null;
  }
...
}

function bacteriumType_change() {
//采用 jquery 获取用户当前选中的 bacteriumTypeId(细菌类型
//标识)
  var bacteriumTypeId = $('#bacteriumTypeId').val();
//拼装 ajax 请求需要的 url
  var drugUrl = "/usercenter/bacteriumType_listXmlDrug.do?
bacteriumTypeId=" + bacteriumTypeId;
//采用 jquery 向服务器端发起 ajax 请求,并从服务端返回的数
//据 data 中解析关联的药品列表数据
  $.get(drugUrl,null,function(data){
   var dom = data.getElementsByTagName("subDrug");
   var drug = document.getElementById("drug");
   var drugHtml = "";
   bacteriumType_clear();
   for(i=0;i<dom.length;i++) {
        var drug_name = (dom [i].getElementsByTagName("
subDrug-name")[0]).firstChild.data;
        var drug_id = (dom [i].getElementsByTagName("
subDrug-id")[0]).firstChild.data;
//生成多个 select 选项,并动态修改 id 为"drug"占位标签
        drugHtml += "<br/>";
        var num = i+1;
        drugHtml += "(" + num + ")" + drug_name + ":" +
          "<select name='drugDrugStatus[" + i + "]'>" +
       "<option value='" + drug_id + "-1'>敏感</option>" +
        "<option value='" + drug_id + "-2'>耐药</option>" +
     "<option value='" + drug_id + "-3'>中介</option>" +
                                          "</select>";
        drugHtml += "<br/>";
   }
   drugHtml += "<br/>";
   drug.innerHTML = drugHtml;
});
...
}
```

由于采用了Ajax方式获取药品列表，因此服务器端也有对应的方法生成药品列表数据，“com.cqnews.bra.action.BacteriumTypeAction”类的listXmlDrug () 方法采用xml的方式返回药品数据：

```
public String listXmlDrug() throws IOException {
        HttpServletRequest request = ServletActionContext.
getRequest();
     HttpServletResponse response = ServletActionContext.
getResponse();
```

```
    // 通过"bacteriumTypeId"取到对应的值
    long bacteriumTypeId = 0;
        String bacteriumTypeId_str = request.getParameter("
bacteriumTypeId");
    if (bacteriumTypeId_str ! = null) {
                    bacteriumTypeId = Long.parseLong
(bacteriumTypeId_str);
    }
    // 清缓存
    response.setHeader("Pragma", "no-cache");
    response.setHeader("Cache-Control", "no-cache");
    response.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("utf-8");
    response.setContentType("text/xml;charset=utf-8");
    // 组装 xml
    Set<Drug> subDrugs = this.bacteriumTypeService.get
(bacteriumTypeId)
                .getDrugs();
    PrintWriter out = response.getWriter();
    out.println("<subDrugs>");
    for (Drug drug : subDrugs) {
        out.println("<subDrug>");
                out.println ("<subDrug-name>" + drug.
getDrugName()
                    + "</subDrug-name>");
        out.println("<subDrug-id>" + drug.getDrugId() + "<
/subDrug-id>");
        out.println("</subDrug>");
    }
    out.println("</subDrugs>");
    return null;
}
```

获取酶的列表与获取药品的列表代码相似。

### 6.2.2 标本保存的业务逻辑的设计

细菌标本和细菌类型、标本类型、科室类型等基础数据存在多对 1 的关系，因此，这部分数据的保存比较方便，可以在 specimen 表 (标本表) 中以外键形式直接保存细菌类型、标本类型、科室类型的 id 值。但由于 1 个细菌标本可能同时和多个药品的某种敏感状态关联，因此必须同时在 specimen_drug 关联表中插入多条状态关联数据，酶也同样存在这样的保存问题。如果不采用 hibernate 等 ORM 框架，必须手动写多条 insert sql 语句完成一个 specimen 对象的保存操作，但采用了 hibernate 框架后，只需持久化 specimen 对象，就可以完成所有关联对象的保存操作了，代码简洁明了，不易出错。

“com.cqnews.bra.action.SpecimenAction” 中的 create () 方法实现标本的保存：

```
public String create() {
    ...
            this.specimenService.save (this.bacterium, this.
bacteriumType,
                    this.specimenType, this.department, this.
drugDrugStatus,
                    this.enzymeEnzymeStatus, this.specimen,
this.appUser);
    ...
}
```

该 create () 方法又调用了 SpecimenService 接口的 save (...) 方法完成了复杂的保存逻辑：

```
/**
 * 保存标本
 *
 * @param bacteriumP
 *          所属细菌实体
 * @param bacteriumTypeP
 *          所属细菌类型实体
 * @param specimenTypeP
 *          所属标本类型实体
 * @param departmentP
 *          所属部门实体
 * @param List
 *          <String> drugDrugStatusP 所选药品状态列表
 * @param List
 *          <String> enzymeEnzymeStatusP 所选酶状态列表
 * @param specimenP
 *          用户端提交的标本数据
 * @param appUserP
 *          录入标本的用户
 * @return String 是否成功
 */
public String save (Bacterium bacteriumP, BacteriumType
bacteriumTypeP,
        SpecimenType specimenTypeP, Department
departmentP,
        List <String > drugDrugStatusP, List <String >
enzymeEnzymeStatusP,
        Specimen specimenP, AppUser appUserP) throws
MyCoreException {
    // 获取 specimen 对象的多个简单关联对象
    Bacterium bacterium = this.bacteriumDao
            .get(bacteriumP.getBacteriumId());
                BacteriumType bacteriumType = this.
bacteriumTypeDao.get(bacteriumTypeP
            .getBacteriumTypeId());
                SpecimenType specimenType = this.
specimenTypeDao.get(specimenTypeP
            .getSpecimenTypeId());
        Department department = this.departmentDao.get
(departmentP
            .getDepartmentId());
        AppUser appUser = this.appUserDao.get(appUserP.
```

```
getAppUserId());
        // 构造需要保存的 specimen 对象
        Specimen specimen = new Specimen();
        specimen.setBacterium(bacterium);
        specimen.setBacteriumType(bacteriumType);
        specimen.setSpecimenType(specimenType);
        specimen.setDepartment(department);
        specimen.setInitDatetime(new java.util.Date());
        specimen.setUpDatetime(new java.util.Date());
        specimen.setCollectDatetime(specimenP.getCollectDat
etime());
        specimen.setAppUser(appUser);
        // 循环构造一组 specimenDrug 对象
        Iterator<String> drugIter = drugDrugStatusP.iterator();
        while (drugIter.hasNext()) {
                String drugStr = drugIter.next();
        Long drugId = Long.parseLong(drugStr.split("-")[0]);
Short drugStatus = Short.parseShort(drugStr.split("-")[1]);
                Drug drug = this.drugDao.get(drugId);
 SpecimenDrugId specimenDrugId = new SpecimenDrugId();
                specimenDrugId.setDrug(drug);
                specimenDrugId.setSpecimen(specimen);
        SpecimenDrug specimenDrug = new SpecimenDrug();
                specimenDrug.setId(specimenDrugId);
                specimenDrug.setDrugStatus(drugStatus);
        specimenDrug.setInitDatetime(new java.util.Date());
        specimen.getSpecimenDrugs().add(specimenDrug);
        }
        // 循环构造一组 specimenEnzyme 对象
        if (enzymeEnzymeStatusP ! = null) {
                                Iterator <String > enzymeIter =
enzymeEnzymeStatusP.iterator();
                while (enzymeIter.hasNext()) {
                        String enzymeStr = enzymeIter.next();
  Long enzymeId = Long.parseLong(enzymeStr.split("-")[0]);
                         Short enzymeStatus = Short.parseShort
(enzymeStr.split("-")[1]);
                Enzyme enzyme = this.enzymeDao.get(enzymeId);
                         SpecimenEnzymeId specimenEnzymeId =
new SpecimenEnzymeId();
                        specimenEnzymeId.setEnzyme(enzyme);
                        specimenEnzymeId.setSpecimen(specimen);
                        SpecimenEnzyme specimenEnzyme = new
SpecimenEnzyme();
                        specimenEnzyme.setId(specimenEnzymeId);
                        specimenEnzyme.setEnzymeStatus
(enzymeStatus);
        specimenEnzyme.setInitDatetime(new java.util.Date());
                                specimen.getSpecimenEnzymes ().add
(specimenEnzyme);
                }
        }
```

```
        // 调用 hibernate 框架的 save(...)方法持久化保存细菌标
//本对象该保存操作同时保存关联的对象,但必须在 Specimen
//对象的 hbm 映射文件中申明这种关联关系
        this.dao.save(specimen);
        ...
}
```

在实体对象 Spcimen 的 hibernate 映射文件 specimen.hbm.xml 中必须明确指定 cascade="save-update" 保存关联的实体对象:

```
...
<hibernate-mapping>
        <class  name = "com.cqnews.bra.model.Specimen"
table="specimen"
        catalog="bra">
        <id name="specimenId" type="java.lang.Long">
                <column name="specimenId" />
                <generator class="native" />
        </id>
        <many-to-one name="bacteriumType"
                                class = "com.cqnews.bra.model.
BacteriumType" fetch="select">
                <column name="bacteriumTypeId" />
        </many-to-one>
...
        <set name="specimenDrugs" inverse="true"
                cascade="save-update">
                <key>
        <column name="specimenId" not-null="true" />
                </key>
                <one -to -many  class = "com.cqnews.bra.
model.SpecimenDrug" />
        </set>
...
        </class>
</hibernate-mapping>
```

### 6.3 "细菌对抗生素敏感性的报表"模块的难点及核心代码分析

"细菌对抗生素敏感性的报表"模块的的设计也存在不少难点（如图 5 所示），比如，报表统计规则复杂，无法直接查询数据库获取最终显示数据。

图 5 "细菌对抗生素敏感性的报表"的查询界面图

由于以上统计数据复杂性的原因，我们编写了 EnzymeBacteriumSens 类对查询结果进行了二次封装，便于 jstl 标签在 jsp 页面中迭代输出：

```
/**
 * 封装用户酶与细菌对药的敏感性数据
 *
 * @author Jason
 *
 */
public class EnzymeBacteriumSens {
    // 药
    private String drugName;
    // 数量(阳性)
    private Long positiveCount;
    // 耐药率(阳性)%
    private Double resistancePositiveRatio;
    // 中介(阳性)%
    private Double mediumPositiveRatio;
    // 敏感(阳性)%
    private Double sensitivePositiveRatio;
    // 数量(阴性)
    private Long negativeCount;
    // 耐药率(阴性)%
    private Double resistanceNegativeRatio;
    // 中介(阴性)%
    private Double mediumNegativeRatio;
    // 敏感(阴性)%
    private Double sensitiveNegativeRatio;
  //每个属性必须生成对应的 getter setter 方法,提供对外访问
    public String getDrugName() {
        return drugName;
    }
    public void setDrugName(String drugName) {
        this.drugName = drugName;
    }
...
}
```

在 “com.cqnews.bra.service.impl.BacteriumServiceImpl” 类的 getBacteriumDrugSensLIst (...) 方法中完成了这个复杂报表数据的生成：

```
/**
 * 生成细菌对药物敏感的报表数据
 *
 * @param startDatetimeP
 *            开始统计时间
 * @param endDatetimeP
 *            结束统计时间
 * @param departmentP
 *            科室
 * @param bacteriumP
 *            细菌
 * @return List<EnzymeBacteriumSens> 封装报表所需数据
 */
public List<EnzymeBacteriumSens> getBacteriumDrugSen
sList(
                    Date startDatetimeP, Date endDatetimeP,
Department departmentP,
        Bacterium bacteriumP) throws MyCoreException {
    List<Drug> drugList = this.drugDao.getAll();
                                List <EnzymeBacteriumSens >
enzymeBacteriumSensList = new ArrayList<EnzymeBacteri
umSens>();
    // 生成报表数据
    // 格式:药 数量(阳性) 耐药率(阳性) 中敏率(阳性) 敏感
// 率(阳性) 数量(阴性) 耐药率(阴性) 中敏率(阴性) 敏感率
// (阴性)
    // 迭代 drug list 生成报表所需 list
    Iterator<Drug> iterDrug = drugList.iterator();
    while (iterDrug.hasNext()) {
        Drug drug = iterDrug.next();
        EnzymeBacteriumSens enzymeBacteriumSens =
new EnzymeBacteriumSens();
        // 构造报表 list
            enzymeBacteriumSens.setDrugName (drug.
getDrugName());
        // 获取全院标本总数量
        Long positiveCount = this.specimenDao
                .getSpecimenTotalCountByHospital
(startDatetimeP,
                endDatetimeP, drug, bacteriumP);
        enzymeBacteriumSens.setPositiveCount
(positiveCount);
        // 获取全院耐药率= (耐药标本数量/ 全院标本总数
//量 ) * 100
        Long resistancePositiveCount = this.specimenDao
                .
getSpecimenTotalCountByHospitalDrugStatus
(startDatetimeP,
        endDatetimeP, drug, bacteriumP, Short
                                .parseShort("2"));
                    Double resistancePositiveRatio = MathUtil.
percentageRound(
    resistancePositiveCount.doubleValue(), positiveCount
                                .doubleValue());
            enzymeBacteriumSens
                            .setResistancePositiveRatio
(resistancePositiveRatio);
        // 获取中敏率(阳性)= (中敏标本数量(阳性)/ 标本
//总数量(阳性) ) * 100
   Long mediumPositiveCount = this.specimenDao.
getSpecimenTotalCountByHospitalDrugStatus
(startDatetimeP,
```

```
            endDatetimeP, drug, bacteriumP, Short
                                                .parseShort("3"));
                    Double mediumPositiveRatio = MathUtil.
percentageRound(
        mediumPositiveCount.doubleValue(), positiveCount
                                    .doubleValue());
            enzymeBacteriumSens.setMediumPositiveRatio
(mediumPositiveRatio);
            // 获取敏感率(阳性)= ( 敏感标本数量(阳性)/ 标本
//总数量(阳性) ) * 100
        Long sensitivePositiveCount = this.specimenDao.
getSpecimenTotalCountByHospitalDrugStatus
(startDatetimeP,
            endDatetimeP, drug, bacteriumP, Short
                                                .parseShort("1"));
                    Double sensitivePositiveRatio = MathUtil.
percentageRound(
        sensitivePositiveCount.doubleValue(), positiveCount
                                    .doubleValue());
            enzymeBacteriumSens
                        .setSensitivePositiveRatio
(sensitivePositiveRatio);
            // 科室统计同理
            Long negativeCount = this.specimenDao
        .getSpecimenTotalCountByDepartment(startDatetimeP,
            endDatetimeP, drug, bacteriumP, departmentP);
            enzymeBacteriumSens.setNegativeCount
(negativeCount);
            // 获取耐药率(阳性)= (耐药标本数量(阳性)/ 标本
//总数量(阳性) ) * 100
        Long resistanceNegativeCount = this.specimenDao.
getSpecimenTotalCountByDepartmentDrugStatus(
                                    startDatetimeP,
endDatetimeP, drug, bacteriumP,
        Short.parseShort("2"), departmentP);
                    Double resistanceNegativeRatio = MathUtil.
percentageRound(
        resistanceNegativeCount.doubleValue(), negativeCount
                                    .doubleValue());
            enzymeBacteriumSens
                        .setResistanceNegativeRatio
(resistanceNegativeRatio);
            // 获取中敏率(阳性)= (中敏标本数量(阳性)/ 标本
//总数量(阳性) ) * 100
Long mediumNegativeCount = this.specimenDao.
getSpecimenTotalCountByDepartmentDrugStatus(
                                    startDatetimeP,
endDatetimeP, drug, bacteriumP,
                    Short.parseShort("3"), departmentP);
                        Double mediumNegativeRatio = MathUtil.
percentageRound(
        mediumNegativeCount.doubleValue(), negativeCount
                                    .doubleValue());
            enzymeBacteriumSens.setMediumNegativeRatio
(mediumNegativeRatio);
            // 获取敏感率(阳性)= ( 敏感标本数量(阳性)/ 标本
//总数量(阳性) ) * 100
        Long sensitiveNegativeCount = this.specimenDao.
getSpecimenTotalCountByDepartmentDrugStatus(
                                    startDatetimeP,
endDatetimeP, drug, bacteriumP,
            Short.parseShort("1"), departmentP);
Double sensitiveNegativeRatio = MathUtil.percentageRound(
        sensitiveNegativeCount.doubleValue(), negativeCount
                                    .doubleValue());
            enzymeBacteriumSens
                        .setSensitiveNegativeRatio
(sensitiveNegativeRatio);
            enzymeBacteriumSensList.add
(enzymeBacteriumSens);
        }
        return enzymeBacteriumSensList;
}
```

从 service 层的以上实现类中可以看出，它并没有包含任何 sql 语句，代码结构非常清晰，那么，该业务层的方法又是怎样访问数据库的呢？

在该类的前面部分，有以下语句，通过接口的方式引入了数据层的 2 个组件完成底层数据的读取操作：

```
...
//@Resource 代表采用注解的方式为 specimenDao 这个成员
//变量注入数据层实现类
@Resource
private SpecimenDao specimenDao;
@Resource
private DrugDao drugDao;
...
```

在 SpecimenDao 这个组件的实现类 SpecimenDaoImpl 中就完成了简单的数据库访问，比如其中的 getSpecimenTotalCount ByHospital (...) 方法就被业务层的 getBacteriumDrugSensList (...) 方法调用来获取全院细菌药物标本总数：

```
...
/**
 * 全院细菌药物标本总数
 *
 * @param startDatetimeP
 *          开始统计时间
 * @param endDatetimeP
 *          结束统计时间
 * @param drugP
 *          药
 * @param bacteriumP
 *          细菌
```

```
 * @return Long 总数
 */
        public Long getSpecimenTotalCountByHospital (Date
startDatetimeP,
                Date endDatetimeP, Drug drugP, Bacterium
bacteriumP) {
String hql = " select count(*) from SpecimenDrug as sd";
      hql += " where sd.id.drug.drugId = ?";
hql += " and sd.id.specimen.bacterium.bacteriumId = ?";
      hql += " and sd.id.specimen.collectDatetime between ?
and ?";
      Object [] params = { drugP.getDrugId (), bacteriumP.
getBacteriumId(),
                  startDatetimeP, endDatetimeP };
      List list = findByHql(hql, params);
      Long totalCount = 0L;
      if (list.size() ! = 0 && null ! = list.get(0)) {
            totalCount = (Long) list.get(0);
      }
      return totalCount;
}
...
```

数据层中大量的 hql 语句（hibernate 查询语言）和具体数据库使用的 SQL 语言很相似，但它们仍然有本质的区别，hql 是对实体对象的查询，SQL 是对数据库表的查询，因此，设计的代码和底层数据库之间就彻底地解除了耦合关系，也就实现了业务系统跨数据库平台部署的设想。

由于篇幅原因，不能对项目涉及的所有功能逐一讲解，如果读者感兴趣，可以在杂志官网下载项目的源程序，并按照文中的思路自行调试和研究。

## 7 结语

从技术角度来说，该“医院检验分析系统”的设计和编码较为科学地遵循了 OOAD 的方法论，其分析过程不仅对其他项目的设计具有很强地指导意义，而且，其编码质量同样具有较高的水平，值得读者研究和学习。

从项目的应用角度来看，该系统已经投入实际运行，对使用单位的日常工作起到了较大的促进作用，减少了医院大量的人力财力开销，提升了生产力。该系统在日后的运行过程中，根据用户反馈和新业务需求，很有可能加入更多实用的功能模块，做得更加完善。

通过一个实际项目的方式抛砖引玉地介绍了 JavaEE 技术怎样做企业级应用系统的开发，文中及项目中难免有很多不足之处，希望得到读者的指正与反馈。

## 参考文献


[1] （英）福勒. 企业应用架构模式. 机械工业出版社，2010，4.

[2] （美）埃克尔. Java 编程思想. 4 版. 机械工业出版社，2007，6.

[3] 刘伟. 设计模式. 清华大学出版社.

[4] 李刚. 轻量级 Java EE 企业应用实战-Struts2 + Spring + Hibernate 整合开发. 2 版. 电子工业出版社.

[5] 蒋彪. JavaEE 是企业级开发首选. 2013.

[6] 安博教育集团著. Hibernate 程序开发. 电子工业出版社.


（收稿日期：2012-05-02）

（上接第 23 页）

```
msoFalse
   Dim i As Integer
   For i = 3 To ActivePresentation.Slides.Count
         ActivePresentation.Slides (i).Shapes (1).TextFrame.
TextRange.Text = ""
   Next i
End Sub
```

至此，系统的代码开发部分就完成了，如果需要增加题目数量，只需要复制相同数量“演讲题目”页即可。

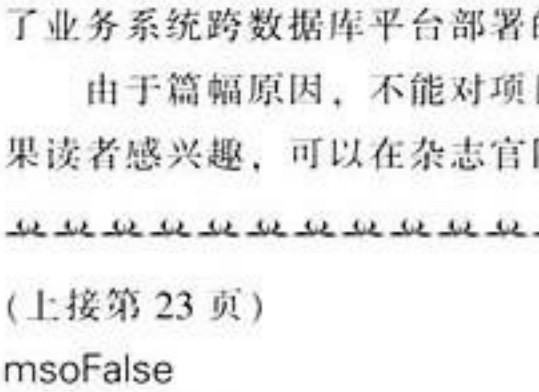



图 4 系统运行结果（一）

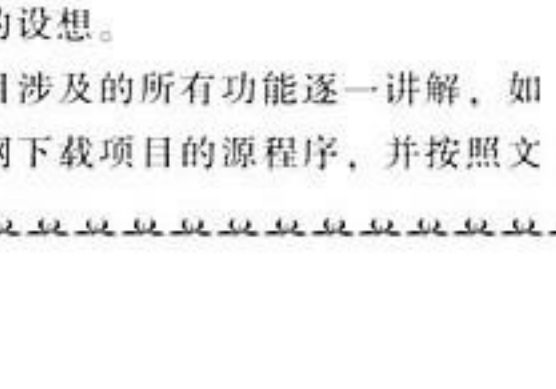



图 5 系统运行结果（二）

（4）程序运行结果。播放此幻灯片，设置好选手的比赛时间，点击“开始选题”按钮，系统开始随机抽题，整个系统的运行结果如图 4、图 5 所示。

## 3 结语

利用 PPT VBA 强大的开发能力，开发出美观、实用的演讲比赛系统。在系统的易用性和演示性方面，PPT VBA 较之其他方法开发具有较大的优势，文中仅仅是给出了一个简单的例子，实现了系统的基本功能，读者可以根据实际需求增加更多功能，如实时评分，选手排名等，开发出一个更加强大的演讲比赛系统。

（收稿日期：2012-03-26）

# Android 应用开发之基于对象数据库 Db4o 的日记账工具

汪永松

摘　要：介绍了将对象数据库 Db4o 的开发包（SDK）移植到 Android 平台，并以 Db4o 数据库为引擎，开发一款日记账簿工具的实践过程。通过此开发案例，读者不仅可以了解 Android 平台移植的模式和要点，而且还可以了解在 Android 平台中开发 Db4o 数据库程序的过程和特点。

关键词：对象数据库；Db4o 开发包；Android 平台；类反射器；谓词

## 1　概述

Android 开发者应该都知道，Android 平台是以 SQLite 引擎作为系统的数据库引擎。SQLite 是一款应用颇广的轻量级关系型数据库（RDB）引擎，支持多数 SQL92 标准。而本文中提到的 Db4o 数据库却是一款嵌入式对象数据库（Object database）。

作为对象数据库，Db4o 以对象作为处理单位（关系型数据库的是记录），其所有的操作接口，例如：添加、更新、删除和查询等，都是以对象为单位。有关 Db4o 数据库的详细说明，读者可以访问 Db4o 的官方网站 http://www.db4o.com/来获取更多的参考信息。

## 2　设计过程

### 2.1　Db4o 开发包

Db4o 开发包分为 Java 和.NET 两个版本，分别支持在 Java 和.NET 平台上的开发。对于 Android 平台而言，只可能接收 Java 版本的 SDK，但这并不表示 Android 平台可以“接纳”任何的 Java 开发包。Android 虚拟机 Dalvik 所支持的字节码格式是 Dex，与普通的 Java 虚拟机所支持的字节码格式不同。如果强制将普通 Java 包加入到 Android 工程，Android 打包工具就会抛出“无法转换到 Dalvik 格式”的错误，如下所示：

```
Conversion to Dalvik format failed: Unable to execute dex: null
Unknown Android Packaging Problem
```

当读者从 Db4o 官方网站（http://www.db4o.com/DownloadNow.aspx）下载到最新的 Db4o Java 版本的开发包，解压之后会发现，Java 版本的开发包又按照 JDK 版本存在 3 个子版本：1.1、1.2 和 5。如图 1 所示。

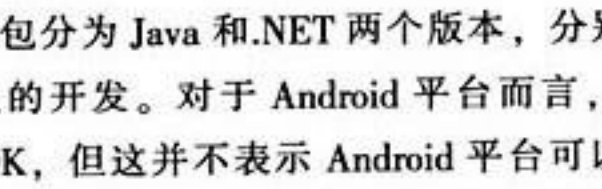



图 1　Db4o 开发包存在多个 Java 子版本

通过测试，Android 平台所支持的 Java 子版本是 JDK5，即文件名形为“db4o-xxx-java5”的包文件。

### 2.2　Db4o 开发包主要功能

Db4o 数据库可以实现对内存中的对象进行存储，对 Db4o 数据库中存储的对象进行查询和删除。通过 Db4o 开发包可以实现对 Db4o 数据库的管理、添加对象、删除对象和对象查询。对于对象存储功能，或许可以通过简单的序列化（Serialize）来将对象保存到存储设备中；但是对于对象查询功能，可就不能这么简单了，特别是当对象数量较多的情况下。Db4o 数据库提供了多种对象查询方式，有兴趣的读者可以参考 Db4o 的相关资料。

#### 2.2.1 数据库管理

不同于 Oracle 等大型数据库，Db4o 数据库的数据存储比较简单，一个数据库文件对应一个数据库（有点类似与 DBF 数据库引擎）。所谓对 Db4o 数据库的管理，就是对 Db4o 数据库文件的管理。

和文件管理一样，对 Db4o 数据库的管理包括创建、删除、移动、复制等。但是在 Android 平台，对于外部存储器（SD 卡）的写入需要在工程清单文件（AndroidManifest.xml）中声明允许写入外部存储器的使用许可（Uses-permission），如以下代码所示：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

使用 Db4o 类的静态方法“openFile”就可以打开指定的 Db4o 数据库，并返回一个对象容器接口（ObjectContainer），为后续对对象的操作提供容器。

使用 Db4o 类实例的“close”就可以关闭当前的数据库。

#### 2.2.2 新增对象

对于日记账簿工具而言，所要新增的内容是支出事项，也就是说每一条支出事项对应一个对象，所以必须定义一个代表支出事项的对象类。有了支出对象，就可以通过 ObjectContainer 接口的“store”方法来存储事项信息。

与一般对象定义不同的是，在 Android 平台，对象如果需要在 Activity 中传递（例如：查询 Activity 需要将对象集合传递给列表 Activity 进行内容显示），则该对象类必须实现 Parcelable 接口。

Parcelable 接口是 Android 所定义的核心机制，在 Activity 之间所传递的数据都经过“打包”的。Parcelable 接口提供了一组写入内容到包和从包中读取数据的方法。

#### 2.2.3 删除对象

使用 ObjectContainer 接口的“delete”方法就可以删除指定对象。

#### 2.2.4 查询对象

使用 ObjectContainer 接口的“query”方法（原生查询）和“queryByExample”方法（按例查询）就可以用于查询指定条件的对象条目，并返回对象集接口（ObjectSet）。

除了原生查询和按例查询，Db4o 还提供了一种 SODA 的查询方式，有兴趣的读者可以参考 Db4o 数据库的有关技术资料。

ObjectSet 继承于 List 接口，所以一旦获取到对象集接口，就可以方便地对其中的对象条目进行处理，例如：显示、汇总、分类等。

### 2.3 界面设计

#### 2.3.1 主界面

图 2 是该日记账簿工具运行的主界面，操作系统是 Android 2.3.6，屏幕分辨率为 480×800 像素。

图 2 日记账簿工具主界面

代码 1 是图 2 所示的主界面的布局定义，其中只包含一张背景图片。

```
//代码 1  主界面的布局定义
//文件名:layout\main_view.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout  xmlns:android = "http://schemas.android.com/
apk/res/android"
   android:orientation="vertical"
            android:layout_width = "match_parent"  android:
layout_height="match_parent">
     <ImageView android:src="@drawable/login_bkg"
       android:layout_width="match_parent"
       android:layout_height="match_parent"/>
</LinearLayout>
```

代码 2 是主界面选项菜单资源的定义，其中包含：添加、浏览和查询 3 个菜单项。

```
//代码 2  主界面选项菜单资源的定义
//文件名:menu\ops_menu.xml

<menu xmlns:android="http://schemas.android.com/apk/res/
//android">
    <item  android:title = "  添 加 记 录 "  android:icon = "
@drawable/append"
        android:id="@+id/mi_add"/>
    <item  android:title = "  浏 览 记 录 "  android:icon = "
@drawable/preview"
        android:id="@+id/mi_view"/>
    <item  android:title = "  查 询 记 录 "  android:icon = "
@drawable/lookup"
        android:id="@+id/mi_look"/>
</menu>
```

#### 2.3.2 记录添加

当用户选择主选项菜单中的“添加记录”项后将会切换到记录添加界面，记录添加完毕之后，记录添加界面将关闭，并提示记录存储是否成功。如图 3 所示。


图 3 记录添加界面

代码 3 是记录添加界面的布局定义，其用到线性布局和表格布局。

```
//代码 3  记录添加界面的布局定义
//文件名:layout\append_view.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout  xmlns:android = "http://schemas.android.com/
apk/res/android"
   android:orientation="vertical" android:layout_width="320sp"

   android:layout_height="fill_parent">
     <TextView android:text="时间戳:"
                android:layout_width = "fill_parent"  android:
layout_height="wrap_content"/>
```

```
    <EditText android:id="@+id/etxt_tsp"
                android:layout_width = "fill_parent" android:
layout_height="wrap_content"/>
    <TextView android:text="备注："
                android:layout_width = "fill_parent" android:
layout_height="wrap_content"/>
    <EditText android:id="@+id/etxt_comments"
                android:layout_width = "fill_parent" android:
layout_height="wrap_content"/>
    <TextView android:text="金额："
                android:layout_width = "fill_parent" android:
layout_height="wrap_content"/>
    <EditText android:id="@+id/etxt_money"
                android:layout_width = "fill_parent" android:
layout_height="wrap_content"/>
    <TableLayout android:stretchColumns="1"
                android:layout_width = "fill_parent" android:
layout_height="wrap_content">
      <TableRow>
                <Button android:id = "@ +id/ebtn_submit"
android:text="确定"
            android:layout_width = "wrap_content" android:
layout_height="wrap_content"/>
                <Button android:id = "@ +id/ebtn_discard"
android:text="取消"
            android:layout_width = "wrap_content" android:
layout_height="wrap_content"/>      </TableRow>
    </TableLayout>
</LinearLayout>
```

### 2.3.3 记录浏览

当用户选择主选项菜单中的“浏览记录”项后将会切换到记录浏览界面，并可以通过按钮“后一条”和“前一条”来顺序浏览所有的记录，如图 4 所示。


图 4　记录浏览界面

可以从图 4 中看出，若当前记录之后有记录，按钮“后一条”才可用；若当前记录之前有记录，按钮“前一条”才可用。

记录浏览界面的布局结构和记录添加界面基本一致，在此不再重复。

### 2.3.4 记录查询

当用户选择主选项菜单中的“查询记录”项后将会切换到记录查询界面，如图 5 中左图所示。


图 5　记录查询界面

记录查询支持模糊匹配，当用户选择图 5 中的按钮“查询”后，将会获取备注（Comments）属性值中包含“超市”文字的所有对象记录。

记录查询界面的布局结构和记录添加界面基本一致，在此不再重复。

### 2.3.5 记录列表

当用户选择记录查询界面中的按钮“查询”后将会切换到记录列表界面。当用户选择列表中某一项时，会在页脚视图中会显示所选项的内容，如图 5 中右图所示。

（1）记录列表界面布局

代码 4 是记录列表界面布局的定义，其主要组件是列表视图。

```
//代码 4　记录列表界面的布局定义
//文件名：layout\report_view.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout  xmlns:android = "http://schemas.android.com/
apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent" android:layout_height="
fill_parent">
     <ListView android:id="@id/android:list"
                android:layout_width = "fill_parent" android:
layout_height="fill_parent"
          android:drawSelectorOnTop="false" />
     <TextView android:id="@id/android:empty"
                android:layout_width = "fill_parent" android:
layout_height="fill_parent"
          android:text="无数据" />
</LinearLayout>
```

（2）列表行视图布局

代码 5 是记录列表界面中行视图的布局定义，其采用表格布局。

```
//代码 5　记录列表界面中行视图的布局定义
//文件名：layout\row_view.xml
<?xml version="1.0" encoding="utf-8"?>
<TableLayout  xmlns:android = "http://schemas.android.com/
apk/res/android"
                android:layout_width = "fill_parent" android:
```

```
layout_height="wrap_content"
    android:orientation="vertical"
    android:stretchColumns="0,1,2">
    <TableRow android:layout_gravity="center_horizontal">
        <TextView android:id="@+id/lab_tsp"
            android:layout_width="wrap_content" android:
layout_height="40sp"
            android:text="时间戳" android:gravity="center"/>
        <TextView android:id="@+id/lab_comments"
            android:layout_width="wrap_content" android:
layout_height="fill_parent"
            android:text="备注" android:gravity="center"/>
        <TextView android:id="@+id/lab_money"
            android:layout_width = "100sp" android:
layout_height="fill_parent"
            android:text="金额" android:gravity="center"/>
    </TableRow>
</TableLayout>
```

## 2.4 功能模块

根据功能之间的耦合度，作者将该日记账簿分为 Activity 和后台功能两块。

### 2.4.1 Activity 组件

（1）主 Activity（JournalBookAct），提供用户与选项菜单的交互，是其他 Activity 的调用者。

（2）记录添加 Activity（AppendRecAct），提供用户与记录添加界面的交互。

（3）记录浏览 Activity（ReviewRecAct），提供用户与记录浏览界面的交互。

（4）记录查询 Activity（LookupRecAct），提供用户与记录查询界面的交互，是记录列表 Activity 的调用者。

（5）记录列表 Activity（ReportRecAct），用于显示查询所得到的记录集信息。

图 6 是该日记账簿工具中 Activity 的交互示意图。


图 6 日记账簿工具 Activity 交互示意图

### 2.4.2 公共定义

（1）支出对象类（Payout），用于代表支出事项，具有时间戳、备注和金额等属性。

（2）支出对象查询谓词（PayoutPredicate），用于具体实现支出对象的查询规则。

（3）对象数据库工具类（OdbUtil），Db4o 数据库 API 的封装 API。

（4）系统工具类（FooSysUtil），提供一些系统功能，例如：获取当前日期时间、日志输出等。

## 2.5 工程清单

代码 6 是该日记账簿工程的清单文件（AndroidManifest.xml）的内容：

```
//代码 6  日记账簿工程清单
//文件名:AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest  xmlns:android = "http://schemas.android.com/apk/
res/android"
    package="foolstudio.demo.db.journalbook"
    android:versionCode="1"
    android:versionName="1.0">
  <uses-sdk android:minSdkVersion="10"/>
   <uses -permission  android:name = "android.permission.
WRITE_EXTERNAL_STORAGE"/>
  <application android:icon="@drawable/icon" android:label="
@string/app_name">
      <activity  android:name = ".JournalBookAct"  android:
label="@string/app_name">
          <intent-filter>
              <action  android:name = "android.intent.action.
MAIN" />
              <category android:name="android.intent.category.
LAUNCHER" />
          </intent-filter>
      </activity>
      <activity  android:name = ".AppendRecAct"  android:
label="@string/act_add"/>
          <activity android:name = ".ReviewRecAct" android:
label="@string/act_view"/>
          <activity android:name = ".LookupRecAct" android:
label="@string/act_look"/>
          <activity android:name = ".ReportRecAct" android:
label="@string/act_report"/>
    </application>
</manifest>
```

代码 6 中，读者可以看出，除了主 Activity 之外，还有 4 个 Activity。

第 7 行声明了允许写外部存储器（SD 卡）的使用许可，这样才能保证程序正常地在 SD 卡中创建 Db4o 数据库。

# 3 开发过程

## 3.1 部署 Db4o 开发包

从 Db4o 官方网站（http://www.db4o.com/DownloadNow.aspx）下载最新 Java 版本的 Db4o 开发包压缩文件。在解压之

后的 lib 文件夹中，可以找到 JDK5 版本的 Jar 文件。

在 Eclipse IDE 中以“添加外部 Jar 包”的方式将该 Jar 文件添加到当前工程中，或者直接改写工程文件夹中的类路径设置文件（.classpath），如代码 7 所示。

```
//代码 7　日记簿工程类路径设置
//文件名:.classpath
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
    <classpathentry kind="src" path="src"/>
    <classpathentry kind="src" path="gen"/>
<classpathentry  kind = "con"  path = "com.android.ide.eclipse.adt.ANDROID_FRAMEWORK"/>
        <classpathentry  kind = "lib"  path = "lib/db4o - 7.4.106.13438-java5.jar"/>
    <classpathentry  kind = "con"  path = "com.android.ide.eclipse.adt.LIBRARIES"/>
    <classpathentry kind="output" path="bin/classes"/>
</classpath>
```

代码 7 中第 6 行，即为添加 Db4o Jar 文件为工程的包文件。实际上，使用 IDE 添加 Jar 文件最终也会添加类路径设置，但 Jar 文件的路径使用的是绝对路径，不便于工程的迁移。

## 3.2　功能实现

### 3.2.1 支出对象定义

代码 8 是支出对象的完整定义，该对象类实例就是 Db4o 数据库所要存储的条目：

```
//代码 8　支出对象定义
//文件名:Payout.java
public class Payout implements Parcelable {
    //类成员
    ...
    //必须要有一个名为 CREATOR 的成员对象,否则无法进
//行 Parcelable 对象通信
 public static final Parcelable.Creator<Payout> CREATOR =
          new Parcelable.Creator<Payout>() {
          public Payout createFromParcel(Parcel in) {
             return new Payout(in);
          }

          public Payout[] newArray(int size) {
             return new Payout[size];
          }
};
//实现 Parcelable 接口
public Payout(Parcel in) { //从包中读取对象属性值
      this.mTsp= in.readString();
      this.mComments = in.readString();
      this.mMoney = in.readDouble();
}
...
    @Override
      public void writeToParcel(Parcel dest, int flags) { //写对
//象到包
          dest.writeString(this.mTsp);
          dest.writeString(this.mComments);
          dest.writeDouble(this.mMoney);
      }
};
```

代码 8 中，支出对象实现了 Parcelable 接口，其中必须要定义一个名为“CREATOR”的全局静态成员，该成员的“createFromParcel”方法定义了从包中生成对象的方法“Payout(in)”。

第 24 行中的“writeToParcel”方法定义了将对象写入到包中的过程。至此，读者可以明白，通过 Parcelable 接口的框架，外部调用组件可以将对象集写入到包中，也可以从包中读出对象集。

### 3.2.2 数据库管理

代码 9 是 Db4o 数据库管理的关键代码，包括打开和初始化数据库文件。

```
//代码 9　Db4o 数据库管理
//文件名:OdbUtil.java
//打开数据库
public ObjectContainer openDB(final String odbName) {
    File file = new File(odbName);
    if(file.exists() == true) { //库文件存在则打开数据库
        return (openDBFile(odbName) );
    }
    else { //如果不存在则初始化
        return(initDB(odbName) );
    }
}
//初始化数据库
private ObjectContainer initDB(final String odbName) {
    File file = new File(odbName);
    if(file.exists() == true) { //库文件已经存在则删除
        if(file.delete() == false) { //删除失败
            return (null);
        }
    }
    return (openDBFile(odbName) );
}
//打开数据库文件
private ObjectContainer openDBFile(final String odbName) {
    //创建一个新的配置实例
    Configuration config = Db4o.newConfiguration();
    //为配置指定特定的反射
    config.reflectWith  (new  JdkReflector  (this.getClass ().
getClassLoader() ) );
    //使用指定的配置打开数据库
     //如不使用指定的配置,在第二次打开数据库文件时会提
```

```
//示有关反射的运行时错误
    return (Db4o.openFile(config, odbName) );
}
```

代码 9 中，必须使用指定的配置来打开数据库，并为其设置特定的类反射器（Reflector），否则在打开已经存在的数据库时，Db4o 平台会抛出类反射器有关的异常。

### 3.2.3 添加记录

代码 10 是在记录添加 Activity 中提交记录添加的关键代码：

```
//代码 10  提交记录添加
//文件名:AppendRecAct.java
//提交添加记录
private void doSubmit() {
    if(submitCheck() == false) { //提交检查
        return;
    }
    //构造支付对象
    Payout payout = new Payout(mTxtTimestamp.getText().
toString().trim(),
            mTxtComments.getText().toString().trim(),
 Double.parseDouble(mTxtMoney.getText().toString().trim()) );
    //调用对象数据库工具类执行添加对象
                OdbUtil.getInstance ().appendObject
(db4oJournalBookAct.DATABASE_NAME, payout);
    FoolUtil.showMsg(this, "对象存储成功！ ");
    this.finish();
}
```

代码 10 中，记录添加 Activity 首先对提交的内容进行检查，然后根据提交的内容生成一个新的支付对象，然后通过调用对象数据库工具类的添加方法将该对象进行存储，其运行效果如图 3 所示。

代码 11 是对象数据库工具类中添加记录对象的方法：

```
//代码 11  添加记录对象
//文件名:OdbUtil.java
//添加对象
public void appendObject(final String odbName, Object obj) {
    //打开数据库
    ObjectContainer odb = openDB(odbName);
    //存储对象
    odb.store(obj);
    //关闭数据库
    closeDB(odb);
}
```

### 3.2.4 浏览记录

代码 12 是记录浏览 Activity 初始化时从 Db4o 数据库中获取对象集的主要代码：

```
//代码 12  初始化对象集
//文件名:ReviewRecAct.java
//初始化数据集
private void initDataSet() {
    Payout proto = new Payout(null, null, 0.0D); //对象原型
        mRecordSet = OdbUtil.getInstance ().getObjects
(db4oJournalBookAct.DATABASE_NAME,
                                        proto);
    mRecordCount = mRecordSet.size();
    ...
    if(mRecordCount > 0) { //初始化显示
        showRecord();
    }
}
//显示记录内容
private void showRecord() {
    Payout payout = mRecordSet.get(mRecordIndex);
    mPrevieTitle.setText("记录 (" + (mRecordIndex+1) + "/"
+ mRecordCount + ")");
    mTxtTimestamp.setText(payout.getTimestamp() );
    mTxtComments.setText(payout.getComments() );
    mTxtMoney.setText (String.valueOf (payout.getMoney ())
);
}
```

代码 12 中第 5 行，记录浏览 Activity 调用了对象数据库工具类的方法来获取数据库中所有对象。获取完毕之后，将记录对象的属性值与显示组件进行绑定。

代码 13 是对象数据库工具类中获取所有对象的方法：

```
//代码 13  获取所有记录对象
//文件名:OdbUtil.java
//获取数据库中指定条件的所有对象(Query By Example)
public ArrayList getObjects (final String odbName, Object
proto) {
    //打开数据库
    ObjectContainer odb = openDB(odbName);
    //查询获得对象集
    ObjectSet objectSet = odb.queryByExample(proto);
    //对象容器
    ArrayList objectDB = new ArrayList();
    while(objectSet.hasNext() ) { //遍历对象集合
        objectDB.add(objectSet.next() );
    }
    //关闭数据库
    closeDB(odb);
    return (objectDB);
}
```

### 3.2.5 查询记录

代码 14 是记录查询 Activity 执行查询行为的主要代码：

```
//代码 14  执行对象查询行为
//文件名:LookupRecAct.java
//执行查询
private void doQuery() {
    //获取查询选择项(目标字段、操作符和查询值)
    String columnName = mSpnColumns.getSelectedItem
```

```
().toString();
        String operator = mSpnOperators.getSelectedItem ().
toString();
        String value = mTxtValue.getText().toString().trim();
    //创建查询谓词实例
    Predicate<Payout> predicate =new JournalBookPredicate
(columnName, operator, value);
        //执行查询操作
                ArrayList recordSet = OdbUtil.getInstance ().
queryObjects(db4oJournalBookAct.DATABASE_NAME,
                                                predicate);
        if(recordSet.size() > 0) { //查询结果不为空
                Intent reportRecIntent = new Intent (this,
ReportRecAct.class);
                reportRecIntent.putParcelableArrayListExtra
(EXTRA_NAME, recordSet);
                this.startActivity(reportRecIntent);
        }
        else {
                FoolUtil.showMsg(this, "查询结果为空,请重试! ");
                return;
        }
}
```

代码 14 中，记录查询 Activity 根据用户的选择来生成支出对象的谓词（Predicate）对象，然后以谓词对象为参数，调用对象数据库工具类的查询方法来获取对象集。

记录查询 Activity 启动记录列表 Activity，并将查询获取到的对象集传递给该 Activity。

代码 15 是对象数据库工具类中查询记录对象的方法。

```
//代码 15  查询记录对象
//文件名:OdbUtil.java
//获取数据库中指定条件的所有对象(Native Query)
        public ArrayList queryObjects (final String odbName,
Predicate predicate) {
        //打开数据库
        ObjectContainer odb = openDB(odbName);
        //查询获得对象集
        ObjectSet objectSet = odb.query(predicate);
        //对象容器
        ArrayList objectDB = new ArrayList();
        while(objectSet.hasNext() ) { //遍历对象集合
                objectDB.add(objectSet.next() );
        }
        //关闭数据库
        closeDB(odb);
        return (objectDB);
}
```

### 3.2.6 对象集传递

代码 16 是在记录列表 Activity 中获取有记录查询 Activity 所传来的对象集的主要代码：

```
//代码 16  记录列表 Activity 接收对象集
//文件名:ReportRecAct.java
//获取参数
Intent intent = getIntent();
ArrayList<Payout> recordSet =
                intent.getParcelableArrayListExtra (LookupRecAct.
EXTRA_NAME);
for(int i = 0; i < recordSet.size(); ++i) {
        addRecord(recordSet.get(i) );
}
```

读者可以看出，代码 16 中的过程和代码 14 中的传递对象集的过程是逆向的。代码 14 中是将对象集放入包中并通过意向对象（Intent）传递，而代码 16 中是从意向对象中获取数据包，并从中提取对象集。

### 3.2.7 查询谓词

代码 17 是查询支付对象的谓词（Predicate）类的定义，该类定义了支付对象的匹配规则，Db4o 数据库引擎以该规则进行对象集的过滤：

```
//代码 17  支付对象查询谓词
//文件名:PayoutPredicate.java
//谓词实例
public class PayoutPredicate extends Predicate<Payout> {
        ...
        @Override
        public boolean match(Payout payout) {
                //时间戳匹配
                        if (mColumnName.compareToIgnoreCase ("
Timestamp") == 0) {
                if(mOperator.compareToIgnoreCase("=") == 0) {
                return (payout.getTimestamp().equals(mValue) );
                        }
        else if(mOperator.compareToIgnoreCase("LIKE") == 0) {
        return (payout.getTimestamp().indexOf(mValue) ! = -1);
                        }
                }
                //备注匹配
                else if (mColumnName.compareToIgnoreCase ("
Comments") == 0) {
                if(mOperator.compareToIgnoreCase("=") == 0) {
                return (payout.getComments().equals(mValue) );
                        }
        else if(mOperator.compareToIgnoreCase("LIKE") == 0) {
        return (payout.getComments().indexOf(mValue) ! = -1);
                        }
                }
                else { //金额匹配
                if(mOperator.compareToIgnoreCase("=") == 0) {
                        return (payout.getMoney () == Double.
parseDouble(mValue) );
```

(下转第 61 页)

# 基于 Excel VBA 的会计单据演示训练系统开发（二）

何 燕

**摘 要**：在财经类专业的会计学课程教学中，为使学生熟练掌握常用原始凭证的填写方法，需要进行模拟训练演示，而传统的手工模拟训练方式存在着一系列缺陷，难以适应会计教学及学习信息化发展方向的需要。为满足这一需要，运用 Excel2007 软件，在 Excel VBA 平台下开发完成原始凭证训练演示系统。

**关键词**：Excel VBA 编程；增值税发票；模拟训练；系统开发

## 1 票面记录读写区制作

### 1.1 读写区分工原理

在支票工作表的右半部分，如图 1 所示，分为 3 块表格区域，每块的表格第一行是支票票面的各填写项目名称。第一块是票面记录库（BA20 格向右下区域），这是所有填写完成并经检验后保存的支票票面记录数据库，数据库的第一行数据是根据样题登记的支票样票记录，并且样票填写顺序号为 0，其后的其他记录依次为 1、2……（略）。第二块是票面记录库读出区（AZ1：BN10 区域），这是从票面记录库中通过程序读出的票面记录，并能通过程序自动填入支票票面。第三块是正在登记的票面记录区（AZ11：BN19 区域），这块区域可通过设置相应函数可记录正在填写的支票各项内容。

### 1.2 主要公式

（1）在 AZ3 格内输入“=IF (BB3="" ,0,COUNTIF (BB3:BB10,BB3))”，这样可统计出票面记录读出区的票面记录行数。

（2）在 AZ13 内输入“=IF (BB13="" ,0,COUNTIF (BB13:BB19,BB13))”，统计正在登记的票面记录行数。

（3）在 BB13 内输入“=IF (C11="" ,"" ,C11)”；在 BC13 内输入：“=IF (OR (C12="" ,E12="" ,G12="") ,"" ,DATE (C12,E12,G12))”；在 BD13 内输入“=IF (C21="" ,"" ,C21)”；在 BE13 内输入“=IF (AA12="" ,"" ,AA12)”；在 BF13 内输入“=IF (AA13="" ,"" ,AA13)”；在 BG13 内输入“=金额数值 (T15:AC15)”；在 BH13 内输入“=IF (M18="" ,"" ,M18)”；在 BI13 内输入“=IF (F26="" ,"" ,F26)”；在 BJ13 内输入“=IF (F27="" ,"" ,F27)”；在 BK13 内输入“=IF (M14="" ,"" ,M14)”；在 BL13 内输入“=IF (M12="" ,"" ,M12)”；在 BM13 内输入“=IF (Q12="" ,"" ,Q12)”；在 BN13 内输入“=IF (T12="" ,"" ,T12)”。BB13：BN13 区域内的函数将支票票面登记情况如实反映出来。

（4）将 BB 列单元格格式设为“文本”格式、BC 列设置为“日期”格式中的“2001—3—14”类型。

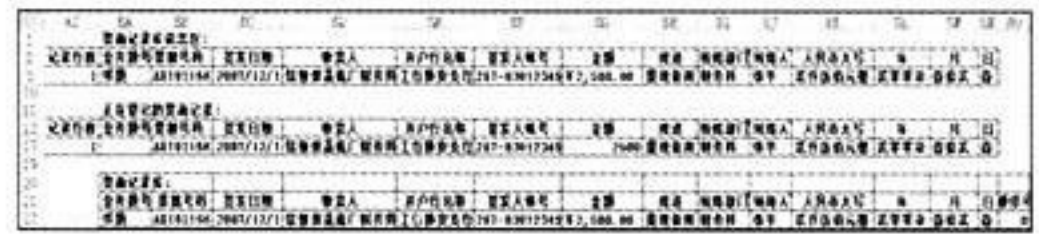


图 1 支票工作表——票面记录读写区
（注：该表中 4~9、14~18 以及 23 行以后因篇幅缘故隐藏）

## 2 增值税发票工作表制作

（1）将支票工作表全表内容复制至增值税发票工作表。

（2）选中增值税发票工作表，并改编左上部分公式。如图 2 所示。

图 2 增值税发票工作表——填写练习部分

1）点击 A2 格，将其内的公式“=训练题! A2”改为“=训练题! AA2”。

2）将 B1 格内的公式“=COUNTA (训练题! A4:A65536)”改为“=COUNTA (训练题! AA4:AA65536)”。

3）将 M3 格的公式“=" 除样题外，本练习共有" &COUNTA (训练题! A4:A65536) &" 条业务题可供选择！可根据业务题或自由填写支票!"”改为“=" 除样题外，本练习共有" &COUNTA (训练题! AA4:AA65536) &" 条业务题可供选择！可根据业务题或自由填写增值税发票!""。

4）将 B4 格内公式“=IF (A4="" ,"" ,IF (A4=0,训练题! B3,INDEX (训练题! B4:B65536,A4)))”改为“=IF (A4="" ,"" ,IF (A4=0,训练题! AB3,INDEX (训练题! AB4:AB65536,A4)))”。

（3）删除工作表中的支票票面及辅助数据部分（选择“删除”命令中的“下方单元格上移”选项），制作增值税专用发票票面如图 2 中 9~27 行所示。

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 30 | | | | | =金额中文大写 (Y21) | | | | | | | |
| 31 | ¥ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | |

图 3　增值税发票工作表——辅助数据部分

（4）将发票号码所在的 W10（W10：AC11 合并）、税务登记号 N12（N12：AC12 合并）及 N23（N23：AB23 合并）的单元格式设置为文本格式。

（5）将开票日期 B11（B11：D11 合并）格设为自定义格式中的“yyyy”年“m”月“d”日格式。

（6）U27（U27：X27 合并）格输入公式“=D23&" 发票专用章""。

（7）价税合计数值 Y21（Y21：AC21 合并）格式为中文会计专用两位小数点格式。

（8）如图 3 所示，在工作表 30~31 行制作增值税发票的辅助数据。

1）B11 格只许输入大于 2001-1-1 的日期数据，设置输入提示框“请输入日期，格式如下：2004-7-6，2008-12-25”，错误警告框“日期应大于等于 2001-1-1!”。

2）F16：H20 区域只允许输入大于零的小数，错误警告框“应输入大于零的数值!”。

3）I16：R20 区域和 T16：AC20 区域只允许输入如 A31：K31 区域的序列，输入提示框“金额前加“¥”!”，错误警告框“只能输入一位整数!”。

4）Y21 格只允许输入大于零的小数，错误警告框“应输入大于零的数值!”（这里不考虑红字发票情况）。

5）定义名称“增值税发票未锁定区域”，引用位置为填写发票时所有可能涉及的单元区域。

6）定义名称“zzsfp 文字不空区域”，引用位置为保存该发票时要求必须填写不能为空的单元区域。

（9）将支票工作表的票面记录读写部分进行改编，改编后效果如图 4、图 5 所示。改编中需注意，AZ~BC 列的表格列标题“记录行数”、“业务题号”、“票据号码”、“日期”保持不变，其他列标题可根据增值税发票票面各项内容重新设置。图 4-5 中部分单元格公式改编如图 6、7、8 所示。

为减少编辑工作量，制作其他原始凭证工作表，可以通过复制支票或增值税发票工作表，再进行改编。如果某种原始凭证各项填写内容只需一行，如收据，可以复制支票工作表后编辑，而某些原始凭证中个别项目填写内容包含多行，如收料单，可复制增值税发票工作表后再进行编辑。

图 4　增值税发票工作表——票面记录读写部分（一）

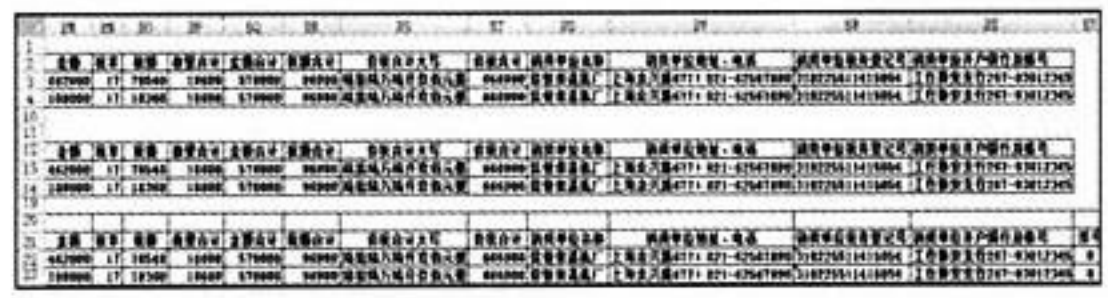


图 5　增值税发票工作表——票面记录读写部分（二）

| | BA | BB | BC | BD | BE | BF | BG | BH |
|---|---|---|---|---|---|---|---|---|
| 13 | 此格通过后面程序填写 | =IF(W10="","",W10) | =IF(B11="","",B11) | =IF(D12="","",D12) | =IF(D13="","",D13) | =IF(N12="","",N12) | =IF(N13="","",N13) | =IF(A16="","",A16) |
| 14 | =IF($BH14="","",BA$13) | =IF($BH14="","",BB$13) | =IF($BH14="","",BC$13) | =IF($BH14="","",BD$13) | =IF($BH14="","",BE$13) | =IF($BH14="","",BF$13) | =IF($BH14="","",BG$13) | =IF(A17="","",A17) |
| 15 | =IF($BH15="","",BA$13) | =IF($BH15="","",BB$13) | =IF($BH15="","",BC$13) | =IF($BH15="","",BD$13) | =IF($BH15="","",BE$13) | =IF($BH15="","",BF$13) | =IF($BH15="","",BG$13) | =IF(A18="","",A18) |
| 16 | =IF($BH16="","",BA$13) | =IF($BH16="","",BB$13) | =IF($BH16="","",BC$13) | =IF($BH16="","",BD$13) | =IF($BH16="","",BE$13) | =IF($BH16="","",BF$13) | =IF($BH16="","",BG$13) | =IF(A19="","",A19) |

图 6　增值税发票 BA13：BX16 区域公式（一）

| | BI | BJ | BK | BL | BM | BN | BO | BP |
|---|---|---|---|---|---|---|---|---|
| 13 | =IF(D16="","",D16) | =IF(E16="","",E16) | =IF(F16="","",F16) | =IF(G16="","",G16) | =IF(金额数值(I16:R16)=0,"",金额数值(I16:R16)) | =IF(S16="","",S16) | =IF(金额数值(T16:AC16)=0,"",金额数值(T16:AC16)) | =IF(F20="","",F20) |
| 14 | =IF(D17="","",D17) | =IF(E17="","",E17) | =IF(F17="","",F17) | =IF(G17="","",G17) | =IF(金额数值(I17:R17)=0,"",金额数值(I17:R17)) | =IF(S17="","",S17) | =IF(金额数值(T17:AC17)=0,"",金额数值(T17:AC17)) | =IF($BH14="","",BP$13) |
| 15 | =IF(D18="","",D18) | =IF(E18="","",E18) | =IF(F18="","",F18) | =IF(G18="","",G18) | =IF(金额数值(I18:R18)=0,"",金额数值(I18:R18)) | =IF(S18="","",S18) | =IF(金额数值(T18:AC18)=0,"",金额数值(T18:AC18)) | =IF($BH15="","",BP$13) |
| 16 | =IF(D19="","",D19) | =IF(E19="","",E19) | =IF(F19="","",F19) | =IF(G19="","",G19) | =IF(金额数值(I19:R19)=0,"",金额数值(I19:R19)) | =IF(S19="","",S19) | =IF(金额数值(T19:AC19)=0,"",金额数值(T19:AC19)) | =IF($BH16="","",BP$13) |

图 7　增值税发票 BA13：BX16 区域公式（二）

| | BQ | BR | BS | BT | BU | BV | BW | BX |
|---|---|---|---|---|---|---|---|---|
| 13 | =金额数值(I20:R20) | =金额数值(T20:AC20) | =IF(E21="","",F21) | =IF(Y21="","",Y21) | =IF(D23="","",D23) | =IF(D24="","",D24) | =IF(N23="","",N23) | =IF(N24="","",N24) |
| 14 | =IF($BH14="","",BQ$13) | =IF($BH14="","",BR$13) | =IF($BH14="","",BS$13) | =IF($BH14="","",BT$13) | =IF($BH14="","",BU$13) | =IF($BH14="","",BV$13) | =IF($BH14="","",BW$13) | =IF($BH14="","",BX$13) |
| 15 | =IF($BH15="","",BQ$13) | =IF($BH15="","",BR$13) | =IF($BH15="","",BS$13) | =IF($BH15="","",BT$13) | =IF($BH15="","",BU$13) | =IF($BH15="","",BV$13) | =IF($BH15="","",BW$13) | =IF($BH15="","",BX$13) |
| 16 | =IF($BH16="","",BQ$13) | =IF($BH16="","",BR$13) | =IF($BH16="","",BS$13) | =IF($BH16="","",BT$13) | =IF($BH16="","",BU$13) | =IF($BH16="","",BV$13) | =IF($BH16="","",BW$13) | =IF($BH16="","",BX$13) |

图 8　增值税发票 BA13：BX16 区域公式（三）

## 3　部分程序

打开 Visual Basic 编辑器，双击工程资源管理器窗口中的 Microsoft Ecxel 对象中的“ThisWorkbook”打开代码窗口，编辑过程“Private Sub Workbook_Open ()”，程序代码如下所示；打开工程资源管理器窗口中的模块 1 代码窗口，编辑公共子程序“Public Sub 训练界面打开 (ws As String)”，程序代码如下：

```
Private Sub Workbook_Open()
Dim i As Integer
Application.ScreenUpdating = False
Sheets("主界面").Visible = True
```

```
For i = 1 To Sheets.Count
    If Sheets(i).Name <> "主界面" Then
        Sheets(i).Visible = False
    End If
Next i
Dim wd As Window
Set wd = ActiveWindow
With wd
    .DisplayGridlines = False
    .DisplayHeadings = False
    .DisplayHorizontalScrollBar = False
    .DisplayVerticalScrollBar = False
    .DisplayWorkbookTabs = False
    .Split = True
    .SplitColumn = 19
    .SplitRow = 44
    .FreezePanes = True
End With
Application.DisplayFullScreen = True
Application.Caption = vbNullChar
 Sheets("主界面").Protect
End Sub
Public Sub 训练界面打开(ws As String)
 Application.ScreenUpdating = False
 Sheets(ws).Visible = True
Sheets("主界面").Visible = False
Dim wd As Window
Set wd = ActiveWindow
With wd
    .ScrollColumn = 1
    .ScrollRow = 1
    .DisplayGridlines = False
    .DisplayHeadings = False
    .DisplayHorizontalScrollBar = False
    .DisplayVerticalScrollBar = False
    .DisplayWorkbookTabs = False
    .Split = True
    .SplitColumn = 36
    .SplitRow = 36
    .FreezePanes = True
End With
Application.DisplayFullScreen = True
Sheets(ws).Protect
 Application.ScreenUpdating = True
End Sub
```

点击“主界面”工作表标签，在主界面工作表中右键点击“支票填写训练”文本框，选中“指定宏”命令，在窗体中点击“新建”按钮，在跳出的代码框中编辑子程序“Sub 按钮 1_Click ()”，这就指定了点击“支票填写训练”文本框时所执行的程序。同法，指定点击“增值税发票填写训练”文本框时所执行的子程序“Sub 按钮 3_Click ()”，其他文本框对应程序也可类似编辑，将“支票”工作表名称改为其他原始凭证工作表名称：

```
Sub 按钮 1_Click()
  Call 训练界面打开("支票")
End Sub
Sub 按钮 3_Click()
 Call 训练界面打开("增值税发票")
End Sub
```

为使原始凭证相应工作表中的凭证票面能自动显示票面记录库读出区的凭证记录，需要在模块 1 中编写公共子程序“Public Sub 细格金额读取 (细格区 As Range, 金额 As Range)”并且针对各原始凭证工作表的“Change”事件分别编程。这里列出相关程序：

```
'本过程将金额数值转化为细格区金额式样读取
Public Sub 细格金额读取(细格区 As Range, 金额 _
As Range)
细格区.Select
Selection.ClearContents
If 金额.Value = "" Then
Exit Sub
End If
Dim colend As Integer, r As Integer
r = 细格区.Row: colend = 细格区.Cells(细格区. _
Count). Column
Dim jg As String, w As Integer, g As Integer
jg = Trim(Str(金额.Value * 100))
w = Len(jg)
Cells(r, colend – w) = "¥"
For g = 0 To w – 1
Cells(r, colend – g) = Mid(jg, w – g, 1)
Next g
End Sub
```

打开 Visual Basic 编辑器，双击工程资源管理器中的“sheet3（支票）”，在右侧打开的代码框中，选择第一个下拉框中的“Worksheet”选项，选择第二个下拉框中的“change”事件，然后在代码框中编程如下：

```
Private Sub Worksheet_Change(ByVal Target As Range)
Application.ScreenUpdating = False
If Target.Cells (1).Column = 53 And _ Target.Cells (Target.
Count).Column = 66 And _
Target.Row = 3 Then
    If Range("BB3").Value <> "" Then
        Application.EnableEvents = False
        Range("C12") = Year(Range("BC3"))
        Range("E12") = Month(Range("BC3"))
Range("G12") = Day(Range("BC3"))
        Range("C21") = Range("BD3")
        Range("M13") = Range("BD3")
        Range("AA12") = Range("BE3")
        Range("AA13") = Range("BF3")
```

```
Call 细格金额读取(Range("T15:AC15"), _ Range("BG3"))
Range("C22") = Range("BG3")
Range("M18") = Range("BH3")
Range("C23") = Range("BH3")
Range("F26") = Range("BI3")
Range("F27") = Range("BJ3")
Range("M14") = Range("BK3")
Range("M12") = Range("BL3")
Range("Q12") = Range("BM3")
    Range("T12") = Range("BN3")
    Range("AA11") = Range("BB3")
    Range("C11") = Range("BB3")
    Application.EnableEvents = True
  End If
End If
Application.ScreenUpdating = True
End Sub
```

增值税工作表“Change”事件编程如下（其他工作表“Change”事件程序略）：

```
Private Sub Worksheet_Change(ByVal Target As Range)
Application.ScreenUpdating = False
If Target.Cells (1).Column = 53 And _ Target.Cells (Target.
Count).Column = 76 And _
Target.Cells(1).Row >= 3 And Target.Cells(1).Row <= 6 And
_ Target.Cells (Target.Count).Row >= 3 And _ Target.Cells
(Target.Count).Row <= 6 Then
  If Range("BB3").Value <> "" Then
    Application.EnableEvents = False
    Range("W10") = Range("BB3").Value
    Range("B11") = Range("BC3").Value
    Range("D12") = Range("BD3").Value
    Range("D13") = Range("BE3").Value
    Range("N12") = Range("BF3").Value
    Range("N13") = Range("BG3").Value
    Range("A16") = Range("BH3").Value
    Range("A17") = Range("BH4").Value
    Range("A18") = Range("BH5").Value
    Range("A19") = Range("BH6").Value
    Range("E16") = Range("BJ3").Value
    Range("E17") = Range("BJ4").Value
    Range("E18") = Range("BJ5").Value
    Range("E19") = Range("BJ6").Value
    Range("D16") = Range("BI3").Value
    Range("D17") = Range("BI4").Value
    Range("D18") = Range("BI5").Value
    Range("D19") = Range("BI6").Value
    Range("F16") = Range("BK3").Value
    Range("F17") = Range("BK4").Value
    Range("F18") = Range("BK5").Value
    Range("F19") = Range("BK6").Value
    Range("G16") = Range("BL3").Value
    Range("G17") = Range("BL4").Value
    Range("G18") = Range("BL5").Value
    Range("G19") = Range("BL6").Value
    Call 细格金额读取(Range("I16:R16"), _ Range("BM3"))
    Call 细格金额读取(Range("I17:R17"), _ Range("BM4"))
    Call 细格金额读取(Range("I18:R18"), _ Range("BM5"))
    Call 细格金额读取(Range("I19:R19"), _ Range("BM6"))
  Range("S16") = Range("BN3").Value
  Range("S17") = Range("BN4").Value
  Range("S18") = Range("BN5").Value
  Range("S19") = Range("BN6").Value
  Call 细格金额读取(Range("T16:AC16"), _ Range("BO3"))
  Call 细格金额读取(Range("T17:AC17"), _ Range("BO4"))
  Call 细格金额读取(Range("T18:AC18"), _ Range("BO5"))
  Call 细格金额读取(Range("T19:AC19"), _ Range("BO6"))
  Range("F20") = Range("BP3").Value
  Call 细格金额读取(Range("I20:R20"), _ Range("BQ3"))
  Call 细格金额读取(Range("T20:AC20"), _ Range("BR3"))
  Range("E21") = Range("BS3").Value
  Range("Y21") = Range("BT3").Value
  Range("D23") = Range("BU3").Value
  Range("D24") = Range("BV3").Value
  Range("N23") = Range("BW3").Value
  Range("N24") = Range("BX3").Value
  Application.EnableEvents = True
 End If
End If
Application.ScreenUpdating = True
End Sub
```

（收稿日期：2012-01-08）

## 技术重塑管理，2013 上海信息化年会在沪举行

近日，“2013 上海信息化年会”在龙之梦万丽酒店举行，正式在上海发布用友新一代高端软件产品——NC6.0。业内卓越流程管理专家、管理技术专家、管理信息化专家、大数据分析专家及多位知名企业的高管齐聚一堂，围绕“转型升级拐点下，中国大型企业如何运用信息技术力量重塑核心竞争力”深入地探讨，引起了来自上海各行业两百多位嘉宾们的共鸣，引发企业管理者们以更多维度和状态审视全球技术浪潮下企业创造性变革与持续发展。

移动互联网、大数据、社交网络、云计算、3D 打印等新技术扑面而来，深刻地影响着整个世界，传统的商业模式、运行模式、管理模式被颠覆，全新的客户关系和市场正逐步覆盖各个商业领域，企业管理已经进入 3.0 时代。拐点当前，中国大型企业的变革需要内外兼备才能获得持续发展——内部企业管理必须向敏捷、精细阶段跃升，提升产品与服务的技术含量，提高运营效益；外部企业需要借助信息技术力量整合产业链资源，完成自我增值到产业链增值，才能洞察市场，卓见未来，明晰方向。

# TreeView 控件基础与综合应用

畅育超

**摘　要**：通过在 VB6.0 中使用 TreeView 树形控件实现员工所在工作部门信息的输入、查询、修改以及 TreeView 控件基本属性和扩展功能的实现。

**关键词**：Treeview 控件；ADO 控件；MSFlexGrid 控件

## 1　引言

随着社会的不断发展，要在大量的各类繁杂的信息中查询所需的信息是非常重要的，而要人性化或者是尽可能地呈现信息本身的一些属性也是很必要的，例如在日常生活中，为了比较准确、人性化地反映一些特殊的信息（单位的组织结构、家族的族谱等），就要用到 TreeView 控件，比较真实、形象的反映出这些特殊信息。为此在 Visual Basic6.0+sp6、Microsoft Access2003 环境下，利用 TreeView 树形控件实现特殊信息的综合处理。

## 2　TreeView 控件常用属性与方法

TreeView 控件显示 Node 对象的分层列表，每个 Node 对象均由一个标签和一个可选的位图组成。TreeView 一般用于显示文档标题、索引入口、磁盘上的文件和目录、或能有效地分层次显示其他特殊种类的信息。为了能更好地使用 TreeView 控件，必须了解它的相关属性，在此介绍其常用的几个属性。

### 2.1　Nodes（节点）

层次结构是一种组织形式，其中的每一部分都与其他部分有亲戚关系。Nodes 是一个节点集合，它里面包含一个或多个 Node 节点，而 Node 节点是 TreeView 控件中的一项，它包含图像与文本。Nodes 既是 TreeView 控件的属性，而它本身也是一个对象。验证此属性的代码如下：

```
Private Sub Command1_Click()'Nodes 属性代码
Dim i As Integer
   Dim strNodes As String
   For i = 1 To TreeView1.Nodes.Count
   strNodes = strNodes & TreeView1.Nodes(i).Index & " " & _
   "Key: " & TreeView1.Nodes(i).Key & " " & _
   "Text: " & TreeView1.Nodes(i).Text & vbLf
   Next i
   MsgBox strNodes & vbLf & " 此说明 Nodes 是一个节点集合,它里面包含一个或多个 Node 节点"
End Sub
```

### 2.2　Root（根）

树形结构的最顶层是根，根也是一个节点，其他节点都是它的分支。一个树只有一个根节点。验证一个树只有个根的代码如下：

```
Private Sub Command2_Click()
Dim anynode As Node
Dim s As String
For Each anynode In TreeView1.Nodes
s = s & anynode.Text &  "的根是:" & anynode.Root.Text & vbLf
Next anynode
MsgBox s & vbLf & "一个树只有一个根节点"
End Sub
```

### 2.3　Parent（父）

Nodes 集合中的每个节点都可以访问根。但要定义树层次结构中的位置，仅知道节点的根是不行的，如果想要知道自己在树结构中的位置，就不能只知道根在哪里，而需要知道更多信息，这既是树形结构中第一个关系：父关系。要成为父节点，必须要有子节点。验证 Parent 属性的代码如下：

```
Private Sub Command3_Click()
Dim mynode As Node
Set mynode = TreeView1.Nodes("R")
If mynode = mynode.Root Then
MsgBox "我是老板！"
Else
MsgBox  "我现在是" &  "" & mynode.Text &  "我要向" & """ & mynode.Parent.Text & """ & "方向努力！"
End If
End Sub
```

从代码可以看出，每个节点对象都有一个与它的父节点有关的 Parent 属性。但是根节点的 Parent 属性与任何节点无关。

### 2.4　Children 属性

要成为一个父节点，必须要有子节点，验证自己是一个父节点的代码如下：

```
Private Sub Command4_Click()
```

```
Dim mynode As Node
Set mynode = TreeView1.Nodes("D1")
If mynode.Children = 0 Then
MsgBox "我是" & mynode.Text & "的" & mynode.Parent.
Text & "属下！"
End If
End Sub
```

从代码可以看到 Children 属性返回的是一个整型数，表示给定节点（父节点）的子节点数量。

### 2.5 Child 属性

刚才讨论过 Children 属性只返回子节点的数量，而 Child 属性返回的是一个真正的节点。但即使一个父节点有多个子节点，Child 属性只能返回给定父节点的后代中的第一个子节点。那如何访问第一个节点以外的其他所有子节点呢，代码如下：

```
Private Sub Command5_Click()
Dim anynode As Node
Dim kidnode As Node
Dim i As Integer
Dim s As String
Dim r As String
Dim t As String
Picture1.Cls
For Each anynode In TreeView1.Nodes
If anynode.Children <> 0 Then
s = anynode.Text & "属下是：" & vbLf
Set kidnode = anynode.Child
anynode.Child.Checked = True
r = s & kidnode.Text
Picture1.Print r
i = kidnode.FirstSibling.Index
While i <> kidnode.LastSibling.Index
t = TreeView1.Nodes(i).Next.Text
TreeView1.Nodes(i).Next.Checked = True
Picture1.Print t
i = TreeView1.Nodes(i).Next.Index
Wend
End If
Next anynode
End Sub
```

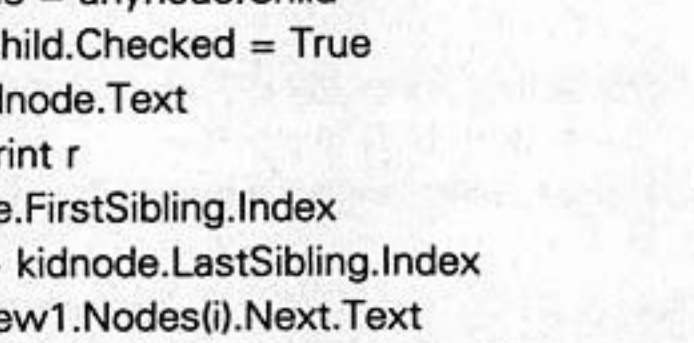



图 1 TreeView 控件属性演示

对于一个给定的节点，它只有一个子节点。与这个子节点共有一个父节点的其他子节点都可以看作是 Next（下一个）节点或 Previous（前一个）节点。所有这些子节点的 Parent 属性有一个相同值，就是 FirstSibling（第一个相邻）和 LastSibling（最后一个相邻）。以上属性演示的效果如图 1 所示。

## 3 TreeView 控件 Nodes 的方法

（1）Add 方法：

1）Add 方法的语法包含下面部分：

Object.Add (Relative,Relationship,Key,Text, Image,Selected Image)

2）参数说明：

① Object 参数：必需的。对象表达式。

② Relative 参数：添加新节点时，其父节点键值 Key。添加根节点时，此项为空。

③ Relationship 参数：新节点的相对位置：

tvwlast—1：新节点位于同级别所有节点之后；

tvwNext—2：新节点位于当前节点之后；

tvwPrevious—3：新节点位于当前节点之前；

tvwChild—4：新节点成为当前节点的子节点。

④ Key：Node 节点关键字（唯一标识符），用于检索 Node 节点。同时也作为其新建子节点的 Relative 值，即新建子节点的 Relative 值就是父节点 Key。

⑤ Text：必需的，Node 节点文本。

⑥ Image：Node 节点位图，是关联 ImageList 控件中位图的索引。

⑦ SelectedImage： Node 节点被选中时呈现的位图索引。

（2）Clear 方法：用于删除 TreeView 控件的所有 Node 节点。

（3）Remove 方法：用于移动 TreeView 控件的节点位置。

## 4 TreeView 控件综合应用

此部分主要介绍 TreeView 控件在实际开发中的具体应用和相关技巧。主效果图如图 2 所示。


图 2 TreeView 控件综合应用

### 4.1 节点的动态编辑

主要介绍了 TreeView 控件中各节点的添加、删除、修改、更新和相关信息的显示。

（1）数据库设计：由于是动态编辑，而 TreeView 控件所显示的数据是保存在数据库里，在此有必要首先说明一下数据

库的设计，由于 TreeView 控件显示的是层次结构数据，那如何来设计此类型的数据库就成为关键，根据需要设计的数据库如图 3 所示。

tsorg : 表

| 部门编号 | 部门名称 | 是否有下级 |
|---|---|---|
| 01 | 公司总部 | Y |
| 0101 | 人事部 | N |
| 0102 | 研发部 | Y |
| 010203 | A小组 | N |
| 010204 | B小组 | N |
| 0103 | 财务部 | N |
| 0104 | 工程部 | Y |
| 010401 | 水电组 | N |
| 010402 | 协调组 | N |
| 0105 | 市场部 | N |

**图 3　数据库设计**

从图 3 中可以看出此表结构中只有 3 个字段，其中“部门名称”意思很明确；而“部门编号”和“是否有下级”两个字段合起来就能准确完整的表示出各部门之间的关系，“部门编号”字段采用长度和值递增的方式进行编码，“是否有下级”字段是个布尔值，如果是“Y”表示其部门是一个父节点，其下还包含一个或多个子节点；如果是“N”表示其部门是个子节点。

（2）功能实现，如图 4 所示。

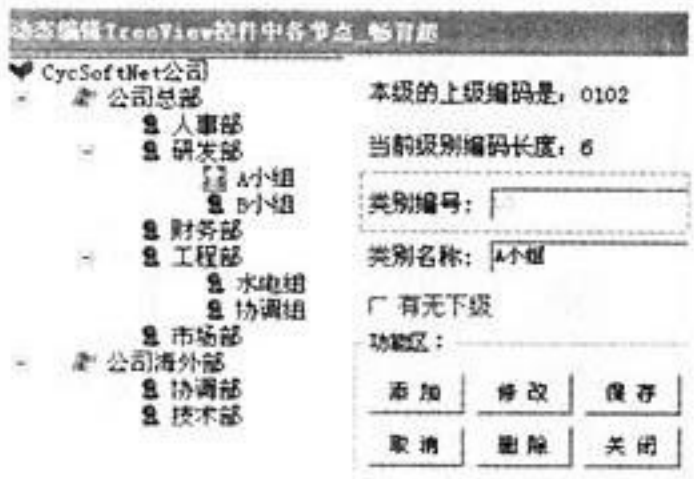



**图 4　功能实现**

为了编号的准确性，在图 4 中的“类别编号”采用自动编号的方式，而且在编辑状态时是只读的，不能更改，在“修改”和“删除”节点时，如果该节点是父节点则不能编辑和删除，否则提出警告，如图 5 所示。

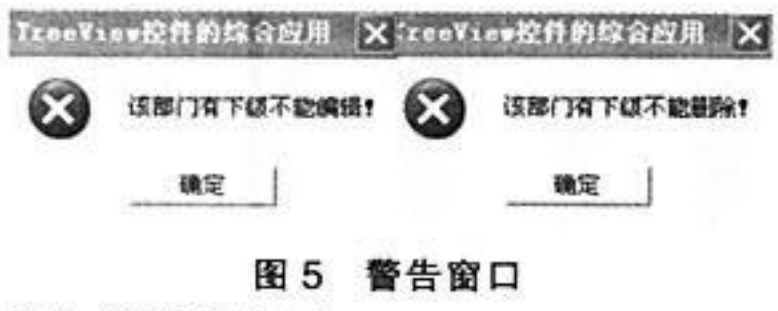



**图 5　警告窗口**

各功能的主要代码如下：

```
Public Sub GetData (ByVal tree As TreeView, ByVal rs As ADODB.Recordset, ByVal Root As String, ByVal L As Integer) ' 定义加载数据到 TreeView 控件的函数
 On Error GoTo jc:
 Dim Node As Node
 Dim str_Key As String
 Dim str_name As String
 Dim M As Integer
 tree.Nodes.Clear
 Set Node = TView.Nodes.Add(, , "Root", Root, 1)
 Node.Expanded = True
 Do Until rs.EOF
   M = Len(Trim(rs.Fields(0))) ' 获得每个字段的长度
   str_Key = rs.Fields(0) ' 部门编号
   str_name = rs.Fields(1) ' 部门名称
   If M = L Then
      If Len(Trim(rs.Fields(0))) = L Then
        Set Node = TView.Nodes.Add ("Root", tvwChild, "K_" & str_Key, str_name, 2) ' 添加父节点
      End If
      Node.Expanded = True
   Else
      If Len(Trim(rs.Fields(0))) = M Then
          Set Node = TView.Nodes.Add ("K_" & Mid(str_Key, 1, M – L), tvwChild, "K_" & str_Key, str_name, 4, 3)' 添加子 ' 节点
      End If
      Node.Expanded = True
   End If
   rs.MoveNext
 Loop
 Exit Sub
jc:
 MsgBox "错误,级别长度不匹配！ ", vbCritical
End Sub
Private Sub TView_Click() 'TreeView 控件单击事件
 If Not TView.SelectedItem Is Nothing Then
    a = Mid (TView.SelectedItem.Key, InStr (1, TView.SelectedItem.Key, "_") + 1)
    Label1.Caption = "本级的上级编码是：" & Left(a, Len(a) – StartLen)
      Label2.Caption = "本级的编码长度是:" & Len(a)
      TxtCode.Text = Right(a, 2)
      TxtName.Text = TView.SelectedItem.Text
      TxtName.Enabled = False
      YN.Visible = False
 End If
 If a <> "" Then
 Command1.Enabled = True
 Command2.Enabled = True
 Command3.Enabled = True
 cmd_cancel.Enabled = True
 End If
End Sub
Private Sub Command1_Click() ' 添加部门信息
 Dim dws As New ADODB.Recordset
 YN.Visible = True
 If TView.SelectedItem Is Nothing Then Exit Sub
 If a = "Root" Then
```

```
    YNADD = True
  Else
    Set rs = conn.Execute ("select * from rsorg where bm='"
& a & "'")
    If rs.Fields(2) = "N" Then
      MsgBox "已经是基层部门,不能再添加！", vbCritical
      Exit Sub
    Else
      YNADD = True ′设置"添加"状态
      Str = rs.Fields(0)
      ′自动编号
      dws.Open "select * from rsorg  where  left (bm,len ('" &
Str & "'))='" & Str & "'  order by bm", conn, adOpenKeyset,
adLockReadOnly
        If dws.RecordCount > 0 Then
          dws.MoveLast
          TxtCode.Text = Format(Val(Right(dws.Fields(0), 2) +
1), "00")
          dws.Close
          Set dws = Nothing
        Else
          TxtCode.Text = "01"
       End If
       TxtName.Enabled = True
       cmd_save.Enabled = True
       TxtName.Text = ""
       TxtName.SetFocus
     End If
End If
End Sub
```

## 4.2 数据查询

本例主要利用 TreeView 控件和 MSFlexGrid 控件联合，实现层次结构数据的查询和显示。如图 6 所示。


图 6 查询数据

在图 6 中，当单击“研发部”时就会查询到该部门下所有人员信息，单击 “研发部”的下属部门“A 小组”时就只显示该组的人员信息。主要代码如下：

```
Private Sub trview_Click() ′单击父节点或子节点时查询相应节
′点的人员信息
If Not trview.SelectedItem Is Nothing Then
a = Mid (trview.SelectedItem.Key, InStr (1, trview.
SelectedItem.Key, "_") + 1)
  If cxrs.State = 1 Then
  cxrs.Close
  End If
    cxrs.Open "select * from rsusertab where left
(rsdepartment,len ('" & a & "')) ='" & a & "' order by
rsdepartment", conn, adOpenKeyset, adLockOptimistic
  If cxrs.RecordCount > 0 Then
  msfgrid.Enabled = True
   With msfgrid
.Rows = 1
Do While Not cxrs.EOF
.Rows = .Rows + 1
.TextMatrix(.Rows – 1, 0) = .Rows – 1
For i = 0 To cxrs.Fields.Count – 1
.Col = i + 1
If IsNull(cxrs.Fields(i)) Then
.TextMatrix(.Rows – 1, i + 1) = Empty
ElseIf InStr(cxrs.Fields(i), "\") <> 0 Then
.TextMatrix(.Rows – 1, i + 1) = Mid(cxrs.Fields(i), InStr(cxrs.
Fields(i), "\") + 1)
Else
.TextMatrix(.Rows – 1, i + 1) = cxrs.Fields(i)
End If
Next i
cxrs.MoveNext
Loop
End With
Else
msfgrid.Enabled = False
End If
 End If
End Sub
```

## 4.3 数据输入

在本例中利用 TreeView 控件实现数据的辅助输入，如图 7 所示。

图 7 辅助输入

为了能直观地看到人员所在部门的具体信息，在图 7 中单击“所在部门”右侧的“双箭头”就出现图 8 所示的 TreeView 控件，在图 8 中，双击所在部门即可获得该人员所在部门的名称。主要代码如下：

```
Private Sub TreeView1_DblClick() ′双击获得部门名称
Dim strbh, strname As String
If Not TreeView1.SelectedItem Is Nothing Then
```

```
    If TreeView1.SelectedItem.Children = 0 Then
       strbh = Mid (TreeView1.SelectedItem.Key, InStr (1,
TreeView1.SelectedItem.Key, "_") + 1)
       strname = TreeView1.SelectedItem.Text
       adduser.Txt_Tree.Text = strname
       adduser.com7.Text = strbh & "\" & strname
     End If
   End If
   Unload Me
End Sub
```


图 8 辅助选项

## 4.4 扩展功能复选框

在此例中，要实现 TreeView 控件扩展功能复选框首先要设置 TreeView 控件的 Checkboxes 值为 True。

(1) 选取父节点下的全部子节点，只需在父节点左侧的复选框单击即可，如图 9 所示。

从图 9 中可看出，当单击“公司总部”时，该部下的所有子部门都被选中，显示在窗体右侧。


图 9 扩展复选框

主要代码如下所示：

```
Private Sub TreeView1_NodeCheck (ByVal Node As
MSComctlLib.Node) Dim anynode As Node
Dim kidnode As Node
Dim i As Integer
Dim s As String
Dim r As String
Dim t As String
Picture1.Cls
For Each anynode In TreeView1.Nodes
If anynode.Checked = True And (anynode.Children <> 0)
Then
s = anynode.Text & "下属是:" & vbLf
Set kidnode = anynode.Child
r = s & kidnode.Text
Picture1.Print r
i = kidnode.FirstSibling.Index
While i <> kidnode.LastSibling.Index
TreeView1.Nodes(i).Checked = True
  t = TreeView1.Nodes(i).Next.Text
TreeView1.Nodes(i).Next.Checked = True
  Picture1.Print t
i = TreeView1.Nodes(i).Next.Index
Wend
End If
Next anynode
TreeView1.Refresh
End Sub
```

(2) 选取父节点和子节点时（即混合选取）时，直接单击该节点即可，如图 10 所示。

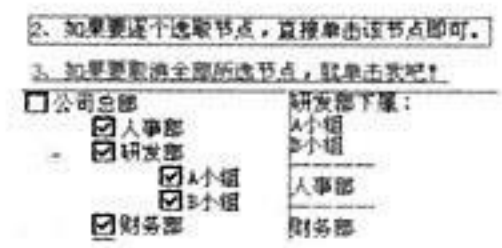



图 10 复选框效果

主要代码如下：

```
Private Sub TreeView1_NodeClick (ByVal Node As
MSComctlLib.Node)
If TreeView1.SelectedItem.Children = 0 And (TreeView1.
SelectedItem.Checked = False) Then
Picture1.Print "————————"
Picture1.Print TreeView1.SelectedItem.Text
TreeView1.SelectedItem.Checked = True
End If
End Sub
```

(2) 取消所有的选择，代码如下：

```
Private Sub Label3_Click() ' 取消所有的选择
Dim anynode As Node
For Each anynode In TreeView1.Nodes
If anynode.Checked = True Then
anynode.Checked = False
End If
Next anynode
Picture1.Cls
End Sub
```

## 5 结语

介绍了 VB6.0 环境下，TreeView 控件的相关属性，并对相关属性的功能进行了代码演示，结合相关控件，讲解了 TreeView 控件在编程过程中的实际应用和技巧。限于篇幅，所有代码以源程序为准。

（收稿日期：2012-03-13）

# 移动办公平台 MAA 上的 HTML5 技术开发与应用

王英华　钟 琪　丁社红

**摘　要**：通过对一个油田移动办公平台的技术原理的介绍，并阐述了功能模块的静态网页源代码转码开发、部署、效果展示等工作，使开发人员可以逐步掌握 HTML5 开发技术。

**关键词**：中原油田；MAA 平台；HTML5 技术；CSS3 技术；XLST 技术；XMLspy 技术

## 1　引言

油田移动办公平台（以下统称为移动办公平台）系统平台是采用国内目前最新的页面转化适配技术，（可灵活地将 JSP、ASPX、ASP、HTML、PHP 等页面转化为标准的 WAP 语言）全称移动代理接入适配中间件服务器，简称 MAA 平台。

油田移动办公平台生产系统软件架构图如图 1 所示。


图 1　油田移动办公平台系统软件架构图

## 2　技术原理

将移动办公平台服务器部署在局域网内部，它与油田内部应用系统部署在一个局域网内，通过局域网利用 HTTP 等多种协议与企业内部 IT 系统（如 OA、合同、邮件、信息门户、新井审批等）进行通信，完成油田业务系统的接入；通过专线或 Internet 网络与运营商的系统进行通信，完成油田接入业务在手机终端上的展现与双向传输。

页面适配方案的工作原理可用图 2 来表示。


图 2　页面适配工作原理

（1）适配服务器抓取数据源的 HTML 页面。

（2）适配服务器将 HTML 文件转换成 XML 元数据。

（3）适配人员以 XML 元数据为基础，通过 UMDP 语言及 XSLT 对元数据进行重构，生成 UMSC 需要的 XML。

（4）适配服务器将 XML 发给移动终端。移动终端对此 XML 进行解析，展现界面。

（5）移动终端发生操作，将指令发送到适配服务器，生成元数据，适配人员通过 UMDP 语言及 XSLT 对元数据进行重构。

（6）适配服务器将接收到的指令进行处理，模拟出数据源可识别的指令发送给数据源。数据源收到指令，完成相关操作。

这实际上就是一个页面抓取→解析→翻译→展现的过程。

## 3　示例说明

以一个油田企业门户为例，如图 3 油田新闻网页界面。


图 3　油田企业门户油田新闻网页界面

（1）在电脑 D 盘安装 D:\Program Files\apache-tomcat-6.0.32，并在 D:\Program Files\apache-tomcat-6.0.32\conf\server.xml 下用记事本打开在</Host>的上一行增加如下代码：

<Context reloadable = "true" docBase = "D:\project\server" path="/htgl"/>。并启动 Tomcat 服务。

（2）在 D：盘部署移动办公平台系统 project 文件夹和 maa-config.xsd 文件。

1）在 D:\project\haike\zyyt\msp\ui 文件夹下，用 XMLspy 编辑工具创建 ui.xml，增加发布到移动办公平台上的油田企业门户 URL 地址。（其他内网应用系统发布到移动办公平台可修改此文件，增加应用系统对应的 URL 即可）。

```
<html>
<head>
    <title>MAA 通用框架</title>
    <meta name="viewport" content="width=device-width, initial-scale=1"/>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>
<body>

    <div data-role="page">
        <div data-role="header">
            <h1>OA 系统</h1>
        </div>
        <div data-role="content">
            <ul data-role="listview" data-dividertheme="d" style="margin-top: 0;">
<li><a href="msp.do?action=bs-view@msp&amp;url=http://ht.zyyt.sinopec.com">合同管理信息系统</a></li>
<li><a href="msp.do?action=bs-view@msp&amp;url=http://leader.zyyt.sinopec.com">局领导专题</a></li>
<li><a href="msp.do?action=bs-view@msp&amp;url=http://oa3.zyyt.sinopec.com">分公司 OA 系统</a></li>
<li><a href="msp.do?action=bs-view@msp&amp;url=http://10.75.1.12/default.aspx">油田主页</a></li>
<li><a href="msp.do?action=bs-view@msp&amp;url=http://oa2.zyyt.sinopec.com">勘探局</a></li>
<li><a href="msp.do?action=bs-view@msp&amp;url=http://oa3.zyyt.sinopec.com">分公司</a></li>
<! --<li><a href="msp.do?action=bs-view@msp&amp;url=http://10.75.0.57/login.aspx">油田论坛</a></li>-->
<li><a href="msp.do?action=bs-view@msp&amp;url=http://lt.zyyt.sinopec.com/login.aspx">油田论坛</a></li>
            </ul>
        </div>
    </div>
</body>
</html>
```

2）在 D:\project\haike\zyyt\msp\config\maa-config.xml 下增加发布到移动办公平台上的油田企业门户对应的各功能网页对应的 URL 地址，如下代码：

```
    <! --油田主页-->
<msp:bs-config source="10.75.1.12" encoding="utf-8">
    <! --首页-->
    <msp:bs -page url = "^http://10.75.1.12/default.aspx $ " responseTemplate="ytzy/zy/index.xsl">
    </msp:bs-page>
    <! --更多-->
    <msp:bs -page url = "^http://10.75.1.12/Lists/List12/infoMore.aspx\?RootFolder" responseTemplate = "ytzy/zy/more.xsl">
    </msp:bs-page>
    <! --详细展示页面-->
    <msp:bs -page url = "^http://10.75.1.12/dwdh/zysybs/Lists/List8/DispForm.aspx" responseTemplate="ytzy/zy/show.xsl">
    </msp:bs-page>
    <msp:bs -page url = "^http://10.75.1.12/Lists/List12/DispForm.aspx" responseTemplate="ytzy/zy/show.xsl">
    </msp:bs-page>
    </msp:bs-config>
    <! --局领导专题配置-->
```

3）用 XMLspy 编辑工具利用 HTML5 技术以及 CSS、XSLT 知识结合油田企业门户网站的油田新闻功能对应的首页：http://10.75.1.12/default.aspx、更多油田新闻功能：http://10.75.1.12/Lists/List12/infoMore.aspx、油田要闻显示功能：http://10.75.1.12/Lists/List12/DispForm.aspx 对应的静态网页源代码进行梳理、选取需要对外发布的功能代码进行分析编写以下 3 个模板代码文件：图 4 是 XMLspy 编辑界面。

图 4 XMLspy 代码编辑主界面

Index.xsl 详细代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/1999/XSL/Transform" version="1.0">
    <! --引入全局公共配置文件-->
    <xsl:import href="./../config.xsl"></xsl:import>
    <! --引入全局公共 URL 配置文件-->
    <xsl:import href="./../baseurl.xsl"></xsl:import>
    <! --引入对当前站点的公共库配置文件-->
    <xsl:import href="./../basemeta.xsl"></xsl:import>
    <! --分析页面-->
    <xsl:template match="/html">
        <html>
```

```
<head>
<title>
<xsl:value-of select="$zy_name"></xsl:value-of>
</title>
<xsl:call-template name="headmeta">
<xsl:with -param name = "mobileinit">1</xsl:with-param>
</xsl:call-template>
<script type="text/javascript">
$(document).ready(function() {
() ;
});
</script>
</head>
<body>
<div data -role = "page" id = "ht_checkDetail" data-theme="b">
<! --顶部导航-->
<div data -role = "header" data -position="fixed" data-backbtn="false" data-theme="b">
<h1 id="usr_name">
<xsl:text></xsl:text>
</h1>
<xsl:call-template name="title_back_link">
</xsl:call-template>
<a href = "{$localurl}layout_zy.xml" data-icon="home" data-iconpos="right" data-direction="reverse" class="ui-btn-right" changeHash="false">退出</a>

<div data-role="header" data-theme="c">
<div data-role="navbar">
<ul>
<li ><a data-theme="b" data-icon="check" href="{$url}={$zy_url}{//div[@id='WebPartWPQ3']/table/tr/td[2]/a/@href}">油田新闻</a></li>
<li ><a data-theme="b" data-icon="grid" href="{$url}={$zy_url}{//div[@id='WebPartWPQ8']/table/tr[2]/td[2]/a/@href}">基层动态</a></li>
<li ><a data-theme="b" data-icon="grid" href="{$url}={$zy_url}{//div[@id='WebPartWPQ9']/table/tr[2]/td[2]/a/@href}">行业新闻</a></li>
</ul>
</div>
</div>
</div>
<div data -role = "content" data -theme="c" data-scroll="true">
<form action = "msp.do" name = "aspnetForm" method="post">
<input type = "hidden" name = "action" value="bs-view@msp"/>
<input type = "hidden" name="url" value="{$zy_url}"/>
<ul data-role="listview" id="shenpi">
<li data -role = "list -divider" style="text-align:left" role="heading" class="ui-li ui-li-divider ui-bar-b">
<a data -theme="b" data-icon="check" href="{$url}={$zy_url}{//div[@id='WebPartWPQ3']/table/tr/td[2]/a/@href}">油田新闻</a>
</li>
<xsl:apply -templates select="//table[@class='ytxw']/tr" mode="ytxw">
</xsl:apply-templates>
<li data -role = "list -divider" style="text-align:left" role="heading" class="ui-li ui-li-divider ui-bar-b">
<a data -theme="b" data-icon="grid" href="{$url}={$zy_url}{//div [@id='WebPartWPQ8']/table/tr[2]/td[2]/a/@href}">基层动态</a>
</li>
<xsl:apply -templates select = "//table [@class ='jcdt']/tr/td/table [1]/tr/td" mode = "base_td">
</xsl:apply-templates>
<li data -role = "list -divider" style = "text -align:left" role = "heading" class="ui-li ui-li-divider ui-bar-b">
<a href = "{$url}={$zy_url}{//div [@id ='WebPartWPQ6']/table/tr [2]/td [3]/a/@href}">党建政工</a>
</li>
<xsl:apply -templates select="//table[@class='djzginfo']/tr" mode="ytxw">
</xsl:apply-templates>
<li data -role = "list -divider" style="text-align:left" role="heading" class="ui-li ui-li-divider ui-bar-b">
<a href = "{$url}={$zy_url}{//div [@id ='WebPartWPQ7']/table/tr [2]/td [3]/a/@href}">职工风采</a>
</li>
<xsl:apply -templates select="//table[@class='zgfcinfo']/tr" mode="ytxw">
</xsl:apply-templates>
<li data -role = "list-divider" style="text-align:left" role="heading" class="ui-li ui-li-divider ui-bar-b">
<a data -theme="b" data-icon="grid" href="{$url}={$zy_url}{//div [@id='WebPartWPQ9']/table/tr[2]/td[2]/a/@href}">行业新闻</a>
```

```
                                    </li>
                                    <xsl:apply  -templates
select="//table[@class='hyxwinfo']/tr" mode="ytxw">
                                    </xsl:apply-templates>
                    <li data-role="list-divider" style="text-align:
left" role="heading" class="ui-li ui-li-divider ui-bar-b">
                                    <a  data -theme = "b"
data -icon = "grid"  href = " {$url} ={$zy_url} {//div  [@id =
'WebPartWPQ10']/table/tr[2]/td[2]/a/@href}">地方新闻</a>
    </li>
                    <xsl:apply-templates select="//table[@class=
'dfxwinfo']/tr" mode="ytxw">
                                        </xsl:apply-templates>
                                    </ul>
                                </form>
                            </div>
    <script type="text/javascript">
    $('#ht_checkDetail').die().live('pageinit', function(event){
                            });
                            makeDownLoadAtt(url){
                                urlInt = url.indexOf("%");
                                (urlInt>0){
                    url = url.replace(/[%]/g,"%25");
                        url = url.replace(/\?/g,"_w_");
                      url = url.replace(/&amp;/g,"_a_");
                                };
                        downloadUrl = "<xsl:value-of select="
$filedown"></xsl:value-of>="+url+"&amp;encoding=gbk";
                                //
                    ("downloadUrl:", downloadUrl);
                                //
                        .open(downloadUrl, '_blank', '');
                                };
                        </script>
                        </div>
                    </body>
                </html>
    </xsl:template>
    //table[@class='ytxw']/tr[1]/td[3]/a
    <xsl:template match="tr" mode="ytxw">
    <xsl:if test="string-length(td[4]/text())>0">
    <li class="ui-li ui-li-static ui-body-c" style="display:
block; ">
                        <a href="{$url}={$zy_url}{td[3]/a/@href}">
    <xsl:value-of select="td[3]/a/text()"></xsl:value-of>
        <xsl:value-of select="td[4]/text()"></xsl:value-of>
                        </a>
                </li>
    </xsl:if>
    </xsl:template>
    <! --基层动态-->
    //table[@class='jcdt']/tr/td/table[1]/tr/td[2]/table/tr/td
    <xsl:template match="td" mode="base_td">
    <xsl:choose>
            <xsl:when test="position()=1">
            <! --
            <img  src = " {$image} ={$zy_url} {a/img/@src}"
alt="图片"></img>
-->
            </xsl:when>
            <xsl:when test="position()=2">
            <xsl:apply -templates  select = "table/tr"
mode="td_tr">
            </xsl:apply-templates>
            <xsl:apply-templates select="//table[@class=
'jcdt']/tr/td/table[2]/tr" mode="td_tr">
            </xsl:apply-templates>
            </xsl:when>
        </xsl:choose>
    </xsl:template>
    <xsl:template match="tr" mode="td_tr">
    <li class="ui-li ui-li-static ui-body-c" style="display:
block; ">
            <a href="{$url}={$zy_url}{td[2]/a/@href}">
    <xsl:value-of select="td[2]/a/text()"></xsl:value-of>
        <xsl:value-of select="td[3]/text()"></xsl:value-of>
            </a>
        </li>
    </xsl:template>
</xsl:stylesheet>
```

more.xsl 详细代码如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet  xmlns:xsl = "http://www.w3.org/1999/XSL/
Transform" version="1.0">
    <! --引入全局公共配置文件-->
    <xsl:import href="./../config.xsl"></xsl:import>
    <! --引入全局公共 URL 配置文件-->
    <xsl:import href="./../baseurl.xsl"></xsl:import>
    <! --引入对当前站点的公共库配置文件-->
    <xsl:import href="./../basemeta.xsl"></xsl:import>
    <! --分析页面-->
    <xsl:template match="/html">
        <html>
            <head>
                <title>
        <xsl:value-of select="$zy_name"></xsl:value-of>
                </title>
            </head>
            <body>
                <div  data  -role  =  "page"  id  =  "
ht_checkDetail" data-theme="b">
                    <! --顶部导航-->
                    <div  data -role = "header"  data -
```

```
position="fixed" data-backbtn="false" data-theme="b">
                        <h1 id="usr_name">
                        <xsl:text></xsl:text>
                        </h1>

                        <a  href = " {$url} ={$zy_url}/
default.aspx"  data -icon = "arrow -l"  data -iconpos = "left"
class="ui-btn-left">后退</a>
                <xsl:call-template name="title_home_link">
                        </xsl:call-template>
            <div data-role="header" data-theme="c">
                <div data-role="navbar">
                                <ul>
                                <xsl:choose>
                                        <xsl:
when test="//span[@class=´ms-sitemapdirectional´]/text()=´
基层动态 ">
                                                <li ><a
data-theme=´b´ data-icon="grid" href="#">基层动态</a></li>
                                </xsl:when>
                        <xsl:when test="//span  [@class=´ms-
sitemapdirectional´]/text()=´ 油田要闻 ">
                                                <li ><a
data-theme=´b´ data-icon="grid" href="#">油田要闻</a></li>
                                </xsl:when>
                        </xsl:choose>

                                </ul>
                        </div>
        </div>
                        </div>
                        <div  data -role = "content"  data -
theme="c" data-scroll="true">
                        <form  action = "msp.do"  name = "
moreForm" method="post">
                        <input  type = "hidden"  name = "
action" value="bs-view@msp"/>
                                <input  type = "hidden"
name="url" value="{$zy_url}/Lists/List12/{//form/@action}"/>
                <ul data-role="listview" id="shenpi">
                                <xsl:choose>
                        <xsl:whentest = "//span  [@class =´ms -
sitemapdirectional´]/text()=´ 基层动态 ">
                        <li data-role="list-divider" style="text-
align:left" role="heading" class="ui-li ui-li-divider ui-bar-b">
                        基层动态
                                                </li>
                                        </xsl:when>
                        <xsl:when test="//span  [@class=´ms-
sitemapdirectional´]/text()=´ 油田要闻 ">
                        <li data-role="list-divider" style="text-
align:left" role="heading" class="ui-li ui-li-divider ui-bar-b">
                        油田要闻
                                                </li>
                                        </xsl:when>
                        <xsl:when test="//span  [@class=´ms-
sitemapdirectional´]/text()=´ 党建政工 ">
                        <li data-role="list-divider" style="text-align:
left" role="heading" class="ui-li ui-li-divider ui-bar-b">
                        党建政工
                                                </li>
                                        </xsl:when>
                        <xsl:when test="//span  [@class=´ms-
sitemapdirectional´]/text()=´ 职工风采 ">
                        <li data-role="list-divider" style="text-align:left"
role="heading" class="ui-li ui-li-divider ui-bar-b">
                        职工风采
                                </li>
                        </xsl:when>
                        <xsl:when  test  =  "//span  [@class  =´ms  -
sitemapdirectional´]/text()=´ 行业新闻 ">
                        <li data-role="list-divider" style="text-align:
left" role="heading" class="ui-li ui-li-divider ui-bar-b">
                        行业新闻
                                                </li>
                                </xsl:when>
                        <xsl:when  test  =  "//span  [@class  =´ms  -
sitemapdirectional´]/text()=´ 地方新闻 ">
                        <li data-role="list-divider" style="text-align:
left" role="heading" class="ui-li ui-li-divider ui-bar-b">
                        地方新闻
                                                </li>

                        </xsl:when>
                                                </xsl:choose>
                                <xsl:apply-templatesselect="
//table[@class=´moreCss´]/tr[position()>1]" mode="base_tr">
                                                </xsl:apply-templates>
                                <xsl:choose>
                                <xsl:when  test = "count  (//tr  [@class =
´cssPager´]/td/table/tr/td)=4">
                                <! --上一页-->
                        <li  class = "ui -li  ui -li -static  ui -body -c"
style="display: block; ">
                        <input  type  =  "button"  data  -theme  =  "b"
name="test" value=" 上一页" data-inline="true" onclick="
dosubmit ({substring-before (substring-after (//tr [@class=
´cssPager´]/td/table/tr/td[2]/input/@onclick,´(´),´)´))"></input>
                                                <
input type="button" data-theme="b" name="test" value="下一
页" data-inline="true" onclick="dosubmit ({substring-before
(substring-after(//tr[@class=´cssPager´]/td/table/tr/td[3]/input/
@onclick,´(´),´)´))"></input>
                                                </li>
                                </xsl:when>
```

```
                <xsl:otherwise>
                    <! --上一页-->
                    <li class="ui-li ui-li-static ui-body-c"
style="display: block; ">
                    <input type="button" data-theme="b"
name="test" value=" 下一页" data-inline="true" onclick="
dosubmit ({substring-before (substring-after (//tr [@class=
'cssPager']/td/table/tr/td[1]/input/@onclick,'('),')')})"></input>
                                                    </li>
                        </xsl:otherwise>
                    </xsl:choose>
<! --构造参数-->
        <div style="display:none">
<xsl:apply-templates select="//input" mode="input">
                    </xsl:apply-templates>
<xsl:apply-templates select="//select" mode="select">
                                        </xsl:apply-templates>
                    </div>
                        </ul>
</form>
                        </div>
<script type="text/javascript">
$('#ht_checkDetail').die().live('pageinit', function(event){
                                });
            function dosubmit(target,argument){
            $("#__EVENTTARGET").val(target);
            $("#__EVENTARGUMENT").val(argument);
                                        //alert("....target...."+
//target+".....argument...."+argument);
                                var url = ' <xsl:value -of
select="//form/@action"></xsl:value-of>';
                                var symbol = '<xsl:value-of
select="//span[@class='ms-sitemapdirectional']/text()"></xsl:
value-of>';
                                if(symbol==' 基层动态 ')
                                {
                                    url = url.replace ('%
u57fa%u5c42%u52a8%u6001',' 基层动态 ');
                                }
                                else if(symbol==' 油田要闻 ')
                                {
                                    url = url.replace ('%
u6cb9%u7530%u8981%u95fb',' 油田要闻 ');
                                }
                        else if(symbol==' 党建政工 ')
                                {
                                    url = url.replace ('%
u515a%u5efa%u653f%u5de5',' 党建政工 ');
                                }
                                else if(symbol==' 职工风采 ')
                                {
                                    url = url.replace ('%
u804c%u5de5%u98ce%u91c7',' 职工风采 ');
                                }
                                else if(symbol==' 行业新闻 ')
                                {
                                    url = url.replace ('%
u884c%u4e1a%u65b0%u95fb',' 行业新闻 ');
                                }

                                else if(symbol==' 地方新闻 ')
                                {
                                    url = url.replace ('%
u5730%u65b9%u65b0%u95fb',' 地方新闻 ');
                                };
                                //alert(".....url....."+url)
        var tempUrl = ' <xsl:value-of select="$zy_url"></xsl:
value-of>'+'/Lists/List12/'+url;
                $('input[name=url]').attr("value",tempUrl);
        //alert(".....value....."+$('input[name=url]').attr("value"));
                $.mobile.changePage ("<xsl:value-of select="$do"
></xsl:value-of>", {
                                    : true,
                                    : 'page',
                        : $('form[name=moreForm]').serialize(),
                                    : 'post'
                                    });
                    };
                makeDownLoadAtt(url){
                    urlInt = url.indexOf("%");
                    (urlInt>0){
            url = url.replace(/[%]/g,"%25");
                url = url.replace(/\?/g,"_w_");
                url = url.replace(/&amp;/g,"_a_");
                                };
            downloadUrl = " <xsl:value -of select = "
$filedown"></xsl:value-of>="+url+"&amp;encoding=gbk";
                                //
            ("downloadUrl:", downloadUrl);
                                //
                .open(downloadUrl, '_blank', '');
                                };
                </script>
                        </div>
                    </body>
                </html>
        </xsl:template>
        <xsl:template match="tr" mode="base_tr">
        <xsl:if test="string-length(td[3]/text())>0">
        <li class="ui-li ui-li-static ui-body-c" style="display:
block; ">
                        <a href="{$url}={$zy_url}{td[2]/a/@href}">
        <xsl:value-of select="td[2]/a/text()"></xsl:value-of>
            <xsl:value-of select="td[3]/text()"></xsl:value-of>
```

```
                    </a>
                </li>
    </xsl:if>
    </xsl:template>
    <xsl:template match="input" mode="input">
    <input type="{@type}" name="{@name}" value="{@value}"
id="{@id}"/>
    </xsl:template>
    <xsl:template match="select" mode="select">
    <select name="{@name}" id="{@id}">
    <xsl:apply-templates select="option" mode="option">
    </xsl:apply-templates>
        </select>
    </xsl:template>
    <xsl:template match="option" mode="option">
    <xsl:choose>
    <xsl:when test="string-length(@selected)>0">
                <option value="{@value}" selected="selected"
><xsl:value-of select="./text()"></xsl:value-of></option>
                </xsl:when>
                <xsl:otherwise>
                <option  value = " {@value}" ><xsl:value -of
select="./text()"></xsl:value-of></option>
                </xsl:otherwise>
            </xsl:choose>
    </xsl:template>
</xsl:stylesheet>
```

show.xsl 详细代码如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet  xmlns:xsl  =  "http://www.w3.org/1999/XSL/
Transform" version="1.0">
    <! --引入全局公共配置文件-->
    <xsl:import href="./../config.xsl"></xsl:import>
    <! --引入全局公共 URL 配置文件-->
    <xsl:import href="./../baseurl.xsl"></xsl:import>
    <! --引入对当前站点的公共库配置文件-->
    <xsl:import href="./../basemeta.xsl"></xsl:import>
    <! --分析页面-->
    <xsl:template match="/html">
        <html>
            <head>
                <title>
            <xsl:value-of select="$zy_name"></xsl:value-of>
                </title>
            </head>
            <body>
                <div  data  -role  =  "page"  id  =  "
ht_checkDetail" data-theme="b">
                    <! --顶部导航-->
                    <div  data -role = "header"  data -
position="fixed" data-backbtn="false" data-theme="b">
                        <h1 id="usr_name">
                            <xsl:text></xsl:text>
                        </h1>
                    <xsl:call-template name="title_back_link">
                        </xsl:call-template>
                    <a  href = "{$url}={//PageRefer}"  data -icon = "
arrow-l" data-iconpos="left"  class="ui-btn-left">后退</a>
                    <xsl:call-template name="title_home_link">
                        </xsl:call-template>
                </div>
                    <div  data -role = "content"  data -
theme="c" data-scroll="true">
                    <form  action = "msp.do"  name = "
aspnetForm" method="post">
                    <input  type  =  "hidden"  name  =  "
action" value="bs-view@msp"/>
    <input type="hidden" name="url" value="{$zy_url}"/>
            <ul data-role="listview" id="shenpi">
                    <xsl:choose>
                    <xsl:when
test="count(//table[@id='MSO_ContentTable']/tr[1]/td[1]/table/
tr/td/table[1])>0">
                <xsl:apply-templates select = "//table  [@id =
'MSO_ContentTable']/tr[1]/td[1]/table/tr/td/table[1]">
                    </xsl:apply-templates>
                    </xsl:when>
    <xsl:when test="count(//table[@id='newstable'])>0">
                <xsl:apply-templates  select = "//table  [@id =
'newstable']"></xsl:apply-templates>
                    </xsl:when>
                    </xsl:choose>
                    </ul>
                    </form>
                </div>
    <script type="text/javascript">
    $('#ht_checkDetail').die().live('pageinit', function(event){

                    });
                </script>
                </div>
            </body>
        </html>
    </xsl:template>
    <! -- img 标签模板-->
<xsl:template match="img">
<div class="pic_01">
        <xsl:choose>
            <xsl:when test="contains(@src,'ttp://')">
        <img src="{$image}={@src}" alt="{@alt}"/>
            </xsl:when>
            <xsl:otherwise>
    <img src="{$imagecache}={$zy_url}{@src}" alt="{@alt}"/>
            </xsl:otherwise>
```

```
        </xsl:choose>
</div>
</xsl:template>
<! -- table 标签模板-->
<xsl:template match="table">
<! --调用模板 tbody-->
<xsl:apply-templates select="tbody"/>
<xsl:apply-templates select="tr"/>
</xsl:template>
<! -- tr 标签模板-->
<xsl:template match="tr">
<xsl:apply-templates/>
<br/>
</xsl:template>
<! -- td 标签模板-->
<xsl:template match="td">
<xsl:apply-templates/>
</xsl:template>
<! -- hr 标签模板-->
<xsl:template match="hr">
<hr/>
</xsl:template>
<! -- ul 标签模板-->
<xsl:template match="ul">
      <br/>
      <xsl:apply-templates/>
      <br/>
</xsl:template>
<! -- li 标签模板-->
<xsl:template match="li">
      <xsl:apply-templates/>
      <br/>
</xsl:template>
<! -- br 标签模板-->
<xsl:template match="br">
<br/>
</xsl:template>
<! -- p 标签模板-->
<xsl:template match="p">
      <xsl:apply-templates/>
      <br></br>
</xsl:template>
<! -- div 标签模板-->
<xsl:template match="div">
      <xsl:apply-templates/>
      <br/>
</xsl:template>
<! -- strong 标签模板-->
<xsl:template match="strong">
<p><font bold="true">
<xsl:value-of select="."></xsl:value-of>
</font></p>
</xsl:template>
<! -- b 标签模板-->
<xsl:template match="b">
<p><font bold="true">
<xsl:value-of select="."></xsl:value-of>
</font></p>
</xsl:template>
</xsl:stylesheet>
```

4）在 PC 机上演示效果

在谷歌浏览器上输入：http://127.0.0.1:8080/htgl，回车，如图 5 所示。


**图 5　PC 机开发测试效果图**

点击图 5“油田主页”按钮进入下一界面如图 6 所示。

**图 6　油田新闻列表图**

点击图 6 红色标注新闻，进入新闻详细内容显示界面如图 7 所示。

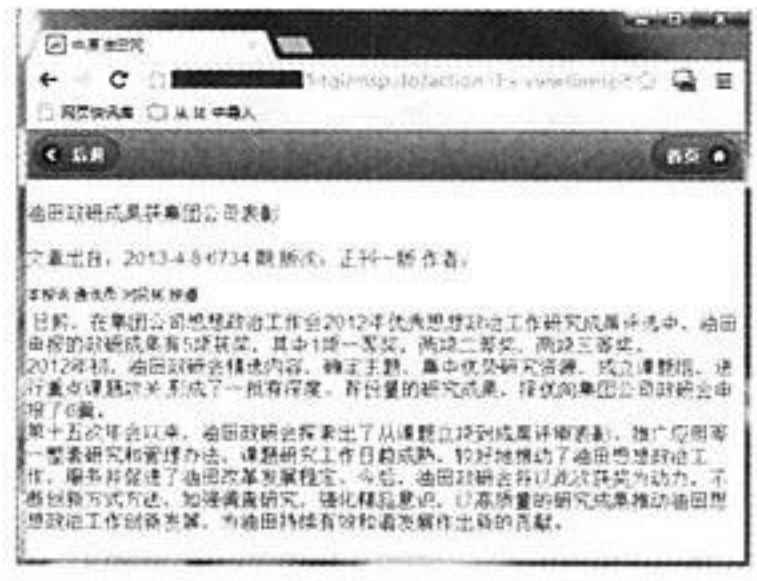


**图 7　新闻详细内容显示界面**

5）移动办公平台生产系统展示效果

以上源代码文件部署到油田移动办公平台生产系统，操作

移动办公平台配置企业门户系统图标及 URL，在手机及手持终端设备的浏览器上输入 3g.zyof.com.cn，进入平台登录界面输入油田用户的 AD 域用户和密码进入移动办公平台“我的应用”主界面如图 8 所示。

图 8　我的应用主界面图

点击图 8“我的应用”主界面的“门户网站”图标进入油田新闻列表如图 9 所示。

9　门户网站油田新闻列表

点击图 9 中标示新闻会出现新闻详细内容界面，如图 10 所示。

图 10　新闻详细内容界面

## 4　结语

通过对移动办公平台的软件结构及技术原理的了解，并经过对移动办公平台的应用系统的部署及对开发技术 HTML5、CSS3、XLST 等技术的培训，就可让开发人员熟练掌握 HTML5 等开发技巧，对油田局域网内的应用系统进行二次转码加工处理、部署，就可使油田领导、员工在任何时间、任何地点通过手持终端设备的浏览器访问移动办公平台与局域网内应用系统交互进行日常工作的办公操作，极大地提高了油田领导、员工的移动办公的工作积极性；同时移动办公平台较强的可延展性能也极大地给予了软件开发人员学习、研究 HTML5 技术的创新动力。

## 参考文献

[1] 申林. 使用 HTML、CSS 和 JavaScript 开发 Android 程序. 电子工业出版社.
[2] 唐俊开. HTML5 移动 Web 开发指南. 电子工业出版社.
[3] 王志刚. HTML5 移动开发即学即用. 电子工业出版社.

（收稿日期：2012-04-12）

（上接第 43 页）

```
            }
    else if(mOperator.compareToIgnoreCase(">") == 0) {
                return  (payout.getMoney () > Double.
parseDouble(mValue) );
            }
    else if(mOperator.compareToIgnoreCase("<") == 0) {
 return (payout.getMoney() < Double.parseDouble(mValue) );
            }
        return (false);
    }
};
```

代码 17 中，自定义谓词类必须继承于 Db4o 的谓词类（Predicate），通过重载谓词类的匹配接口（match）来定制支付类对象的匹配规则。

## 4　结语

通过一个日记账簿工具阐述了在 Android 平台进行 Db4o 数据库应用方面的开发过程。不仅让读者了解外部开发包引入到 Android 平台的方法和要点，而且还领会到对象数据库 Db4o 的主要功能以及在 Android 平台中的使用要点。

以下是应该注意的要点：

（1）并不是所有 Java 开发包都能够纳入 Android 平台，至少能够支持 Dex 字节码格式。

（2）在 Eclipse IDE 中通过菜单添加外部 Jar 包，引用的是该Jar 文件的绝对路径，而直接编辑工程的类路径配置文件可以引用 Jar 文件的相对路径，更利于工程文件夹的迁移。

（3）在 Android 平台，在 SD 卡中创建 Db4o 数据库需要在工程清单文件中声明允许写外部存储器的使用许可。

（4）为了保证正常打开已经存在的 Db4o 数据库，在打开数据库文件时，必须指定配置，并为配置指定特定的类反射器。

（5）为了满足对象能够在 Activity 组件之间进行传递，对象类必须实现 Parcelable 接口。

（收稿日期：2012-01-24）

# 基于 UDP 协议的点对点语音通信软件设计

赵常寿 张玉忠 樊 蓉

**摘 要**：基于 Windows 的低层音频服务、UDP 用户数据报协议，开发了一个简单的点对点语音通信程序，介绍了语音通信原理并给出了相关功能的实现代码。

**关键词**：录音；放音；回调线程；UDP 协议

## 1 引言

随着计算机网络日益普及，人们通过网络进行交流越来越普遍，出现了一系列支持语音通信的软件，比如腾讯 QQ、阿里旺旺、微软 MSN、Skype 等。这些软件都功能完善、相对独立，有时希望自己的软件中也具有语音通信功能。下文通过一个具体实例讨论语音通信的实现方法，硬件需要一块声卡和一个耳麦，程序在 Windows XP SP3 和 VC++6.0 下编译通过，测试效果良好。

## 2 软件结构

要实现点对点语音通信，只要针对一点实现语音的实时采集、处理、播放，同时能进行网络发送和接收，这样两点一连便可通话。对于前者，采用 Windows 的低层音频服务，利用其中的回调机制，在应用程序与设备驱动程序的相互配合下，完成录音和放音功能。在点对点网络传输方面，选择 UDP 用户数据报协议，利用 UDP 连接，在采集语音后即时传给网络，同时接收网络传来的语音，这样就实现点对点语音通信，其通信原理如图 1 所示。

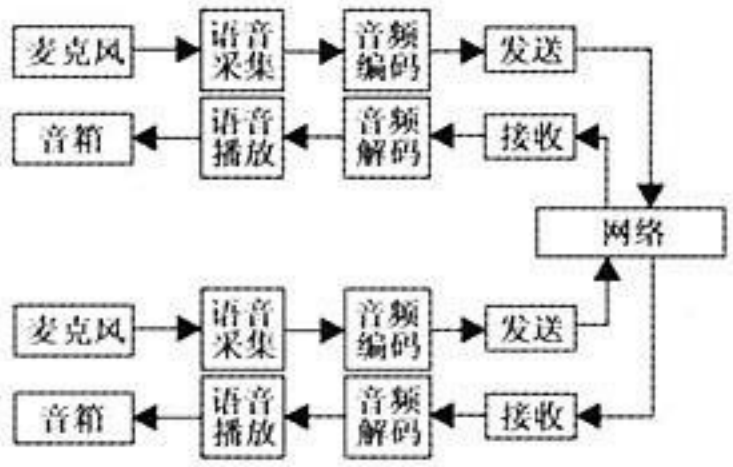



图 1 点对点语音通信原理图

## 3 功能实现

### 3.1 语音采集

语音采集过程是：打开录音设备，开启录音缓存，启动录音，录音处理，停止录音，关闭录音缓存，关闭录音设备。

#### 3.1.1 打开录音设备

打开录音设备需要 API 函数 waveInOpen，函数原型如下：

```
MMRESULT waveInOpen
(
    LPHWAVEIN phwi,    //设备句柄
    UINT uDeviceID,    //设备标识
    LPWAVEFORMATEX pwfx, //录音格式
     DWORD dwCallback, //处理 MM_WIM_*** 消息的回调
//函数、窗口或线程
    DWORD dwCallbackInstance,//用户实例数据
    DWORD fdwOpen    //设备打开方式
);
```

其中，第 1 个参数是设备句柄指针，如果 fdwOpen 为 WAVE_FORMAT_QUERY 则此参数为 NULL；第 2 个参数是设备标识，一般为 WAVE_MAPPER；第 3 个参数是录音格式 WAVEFORMATEX 指针，定义如下：

```
typedef struct
{
    WORD wFormatTag;        //数据格式
    WORD nChannels;         //声道数
    DWORD nSamplesPerSec;       //采样频率
    DWORD nAvgBytesPerSec;      //每秒数据量
    WORD nBlockAlign;       //块对齐字节数
    WORD wBitsPerSample;    //样本大小
    WORD cbSize;
}    WAVEFORMATEX;
```

第 4 个参数是处理消息的回调函数、事件句柄、窗口句柄或线程标识；第 5 个参数是传递给回调机制的用户实例数据；第 6 个参数是设备打开方式。

程序中录音格式设置为：WAVE_FORMAT_PCM 数据格式、单声道、22050Hz 采样频率、16 位样本大小。回调机制采用线程模式，设备打开方式为 CALLBACK_THREAD，程序代码如下：

```
//打开录音设备
BOOL CWaveIn::OpenDevice()
{
    if(m_bOpenDevice)
        return TRUE;
```

```
//设置录音格式
WAVEFORMATEX wf;
wf.wFormatTag=WAVE_FORMAT_PCM;
wf.nChannels=1;
wf.nSamplesPerSec=22050;
wf.nAvgBytesPerSec=44100;
wf.nBlockAlign=2;
wf.wBitsPerSample=16;
wf.cbSize=0;
    //查询设备是否支持该录音格式
    m_mmr  =waveInOpen  (0,WAVE_MAPPER,&wf,0,0,
WAVE_FORMAT_QUERY);
    if(m_mmr)
        return FALSE;
  //以线程回调方式打开设备,m_dwAudioInId 为录音线程 ID
    m_mmr  =waveInOpen  (&m_hIn,WAVE_MAPPER,&wf,
m_dwAudioInId,0,CALLBACK_THREAD);
    if(m_mmr)
        return FALSE;
    m_bOpenDevice=TRUE;
    return TRUE;
}
```

### 3.1.2 开启录音缓存

打开设备后，开始录音前要设置录音缓存，用 API 函数 waveInPrepareHeader 为录音设备准备缓存，用 API 函数 waveInAddBuffer 将缓存添加给录音设备。录音缓存使用的数据结构是 WAVEHDR，定义如下：

```
typedef struct {
LPSTR lpData;              //缓存指针
DWORD dwBufferLength;//缓存长度
DWORD dwBytesRecorded; //已录音的字节长度
DWORD dwUser;              //用户数据
DWORD dwFlags;             //缓存信息标识
DWORD dwLoops;         //播放循环次数
struct wavehdr_tag * lpNext;   //保留
DWORD reserved;            //保留
} WAVEHDR;
```

其中，主要用到第一个和第二个参数，其他参数设置为 0。

程序中设置了 5 块录音缓存，组成一个缓存队列提供给录音设备循环使用，程序代码如下：

```
//开启录音缓存
BOOL CWaveIn::OpenBuffer()
{
    if(m_bBuffer)
        return TRUE;
    int i;
    // WAVEHDR 数组
    m_pHdr=new WAVEHDR[NUM_BUF];
    for(i=0;i<NUM_BUF;i++)
    {
        //设置录音缓存
        ZeroMemory(m_pHdr+i,sizeof(WAVEHDR));
    m_pHdr[i].lpData=new char[SIZE_AUDIO_FRAME];
        m_pHdr[i].dwBufferLength=SIZE_AUDIO_FRAME;
        //为录音设备准备缓存
        m_mmr=waveInPrepareHeader  (m_hIn,m_pHdr+i,
sizeof(WAVEHDR));
        if(m_mmr)
            return FALSE;
        //将缓存添加给录音设备
        m_mmr=waveInAddBuffer(m_hIn,m_pHdr+i,sizeof
(WAVEHDR));
        if(m_mmr)
            return FALSE;
    }
    m_bBuffer=TRUE;
    return TRUE;
}
```

### 3.1.3 录音处理

录音处理模块用来响应录音设备发来的消息，相关消息如下：

（1）MM_WIM_OPEN：打开录音设备时的消息。

（2）MM_WIM_CLOSE：关闭录音设备时的消息。

（3）MM_WIM_DATA：当缓存已满或者停止录音时的消息。

程序中采用线程回调模式实现消息处理，在 MM_WIM_DATA 消息里，对缓存进行重新分配，实现录音缓存循环使用，程序代码如下：

```
//录音消息处理线程
DWORD WINAPI CWaveIn::acc_thread(LPVOID lpParameter)
{
    CWaveIn  *p=(CWaveIn*)lpParameter;
    BOOL bReset=FALSE;
    BYTE  buffer[SIZE_AUDIO_FRAME];
    MSG  msg;
    while(GetMessage(&msg,0,0,0))
    {
        switch(msg.message)
        {
        case WM_USER+10:      //允许循环使用录音缓存
            bReset=FALSE;
            break;
        case WM_USER+20:      //禁止循环使用录音缓存
            bReset=TRUE;
            break;
        case MM_WIM_OPEN:     //打开录音设备时的消息
            break;
        case MM_WIM_CLOSE:    //关闭录音设备时的消息
            break;
        case MM_WIM_DATA:     //当缓存已满或者停止录
//音时的消息
        WAVEHDR *pHdr=(WAVEHDR*)msg.lParam;
```

```
                //清除录音缓存
                waveInUnprepareHeader    ((HWAVEIN)msg.
wParam,pHdr,sizeof(WAVEHDR));
            CopyMemory(buffer,pHdr->lpData,SIZE_AUDIO_
FRAME);
                if(! bReset)
                {
                    //为录音设备准备缓存
                    waveInPrepareHeader ((HWAVEIN)msg.
wParam,pHdr,sizeof(WAVEHDR));
                    //将缓存添加给录音设备
                        waveInAddBuffer   ((HWAVEIN)msg.
wParam,pHdr,sizeof(WAVEHDR));
                }
                //将语音数据发送到网络
                p->SendAudio(buffer,SIZE_AUDIO_FRAME);
                break;
            }
        }
        return msg.wParam;
}
```

#### 3.1.4 相关录音控制函数

（1）启动录音函数

```
MMRESULT waveInStart( HWAVEIN hwi );
```

（2）停止录音函数

```
MMRESULT waveInReset( HWAVEIN hwi );
```

（3）关闭录音设备函数

```
MMRESULT waveInClose( HWAVEIN hwi );
```

（4）准备缓存函数

```
MMRESULT waveInPrepareHeader(HWAVEIN hwi, LPWAVE
HDR pwh, UINT cbwh );
```

（5）添加缓存函数

```
MMRESULT waveInAddBuffer(HWAVEIN hwi, LPWAVEHDR
pwh, UINT cbwh );
```

（6）清除缓存函数

```
MMRESULT waveInUnprepareHeader ( HWAVEIN hwi,
LPWAVEHDR pwh, UINT cbwh);
```

这些函数的具体说明读者可以参见 MSDN 帮助，在此略过。

### 3.2 语音播放

语音播放过程是：打开放音设备，开启放音缓存、放音，放音处理，关闭放音设备。相对于录音来说，放音简单多了，用到的函数主要是以下几个：

（1）打开放音设备

```
MMRESULT waveOutOpen(
    LPHWAVEOUT phwo,
    UINT uDeviceID,
    LPWAVEFORMATEX pwfx,
    DWORD dwCallback,
    DWORD dwCallbackInstance,
    DWORD fdwOpen
);
```

（2）为放音设备准备内存块

```
MMRESULT waveOutPrepareHeader (HWAVEOUT hwo,
LPWAVEHDR pwh, UINT cbwh);
```

（3）写数据（放音）

```
MMRESULT waveOutWrite (HWAVEOUT hwo,
LPWAVEHDR pwh, UINT cbwh);
```

相应也有 3 个消息，用法跟录音的类似，下面列出主要代码，完整代码可参见源程序。

#### 3.2.1 打开放音设备

```
//打开放音设备
BOOL CWaveOut::OpenDevice()
{
    if(m_bOpenDevice)
        return TRUE;
    //设置放音格式
    WAVEFORMATEX wf;
    wf.wFormatTag=WAVE_FORMAT_PCM;
    wf.nChannels=1;
    wf.nSamplesPerSec=22050;
    wf.nAvgBytesPerSec=44100;
    wf.nBlockAlign=2;
    wf.wBitsPerSample=16;
    wf.cbSize=0;
  //查询设备是否支持该放音格式
    m_mmr  =waveOutOpen  (0,WAVE_MAPPER,&wf,0,0,
WAVE_FORMAT_QUERY);
    if(m_mmr)
        return FALSE;
 //以线程回调方式打开设备,m_dwAudioOutId 为放音线程 ID
    m_mmr  =waveOutOpen  (&m_hOut,WAVE_MAPPER,
&wf,m_dwAudioOutId,0,CALLBACK_THREAD);
    if(m_mmr)
        return FALSE;
    waveOutSetVolume(m_hOut,0xFFFFFFFF);
//音量设为最大
    num=0;
    m_bOpenDevice=TRUE;
    return TRUE;
}
```

#### 3.2.2 开启放音缓存、放音

```
//开启放音缓存、放音
BOOL CWaveOut::Play(BYTE *data,int size)
{
    if(BufferNum()>5)    //放音设备缓冲区满,丢弃后续帧
        return TRUE;
    BYTE *p;
    LPWAVEHDR pHdr=new WAVEHDR;
```

```
if(! pHdr)
        return FALSE;
p=new BYTE[size];
if(! p)
        return FALSE;
CopyMemory(p,data,size);
ZeroMemory(pHdr,sizeof(WAVEHDR));
pHdr->lpData=(char*)p;
pHdr->dwBufferLength=size;
    //准备放音缓存
      m_mmr =waveOutPrepareHeader (m_hOut,pHdr,sizeof
(WAVEHDR));
      if(m_mmr)
            return FALSE;
    //放音
    m_mmr=waveOutWrite(m_hOut,pHdr,sizeof(WAVEHDR));
      if(m_mmr)
            return FALSE;
      BufferAdd();

      return TRUE;
}
```

### 3.2.3 放音处理

```
//放音处理线程
DWORD WINAPI CWaveOut:acc_thread(LPVOID lpParameter)
{
    CWaveOut *p=(CWaveOut*)lpParameter;

    MSG msg;
    while(GetMessage(&msg,0,0,0))
    {
        switch(msg.message)
        {
        case MM_WOM_OPEN:
            break;
        case MM_WOM_CLOSE:
            break;
        case MM_WOM_DONE:
        WAVEHDR *pHdr=(WAVEHDR*)msg.lParam;
                waveOutUnprepareHeader    ((HWAVEOUT)
msg.wParam,pHdr,sizeof(WAVEHDR));
                p->BufferDec();
                delete []pHdr->lpData;
                delete pHdr;
                break;
            }
      }
      return msg.wParam;
}
```

### 3.2.4 关闭放音设备

```
//关闭放音设备
BOOL CWaveOut::CloseDevice()
{
      if(! m_bOpenDevice)
            return TRUE;
      //关闭放音设备
      m_mmr=waveOutClose(m_hOut);
      if(m_mmr)
            return FALSE;
      m_hOut=0;
      m_bOpenDevice=FALSE;
      return TRUE;
}
```

## 3.3 网络传输

网络传输模块完成语音数据的发送和接收，传输协议采用UDP用户数据报协议。通过从CSocket类派生一个新类，在派生类里重载虚函数OnReceive，在OnReceive函数里调用数据接收代码，在录音处理线程里调用数据发送代码。相关API函数是ReceiveFrom和SendTo。

ReceiveFrom函数原型如下：

```
int ReceiveFrom(
      void* lpBuf,
      int nBufLen,
      CString& rSocketAddress,
      UINT& rSocketPort,
      int nFlags = 0 );
```

其中，第1个参数是接收缓冲区，第2个参数是缓冲区长度，第3个参数是发送方IP地址，第4个参数是发送方端口号。

SendTo函数原型如下：

```
int SendTo(
      const void* lpBuf,
      int nBufLen,
      UINT nHostPort,
      LPCTSTR lpszHostAddress = NULL,
      int nFlags = 0 );
```

其中，第1个参数是发送缓冲区，第2个参数是发送缓冲区长度，第3个参数是接收方端口号，第4个参数是接收方IP地址。

### 3.3.1 接收数据代码

```
//接收数据
void CUdp::OnReceive(int errcode)
{
      ((CAccDlg*)dlg)->ReceiveAudio();
}
void CAccDlg::ReceiveAudio()
{
    //接收数据
        if(m_Udp.ReceiveFrom(bufReceive,SIZE_AUDIO_PAC
KED,chIPin,uiPort)==SOCKET_ERROR)
            return;
```

（下转第86页）

# 24 位色 BMP 转换 8 位色 BMP

江 洪

**摘 要**：使用 VC6.0 开发了一个可以实现 24 位色 BMP 转换 8 位色 BMP 功能的程序。程序使用比较常用的流行色算法生成调色板，加入了误差扩散处理颜色差异，可以生成 RLE8 压缩或非压缩的 BMP 文件，具有一定的实用价值。

**关键词**：BMP 文件；调色板；流行色；误差扩散；RLE8 压缩

## 1 引言

BMP 文件，即点位图文件，是一种很常见的图像文件格式，能够被大多数软件所支持，在很多场合都有广泛的应用。

BMP 文件按照颜色位数不同，有 1、4、8、16、24、32 几种。其中 24 位 BMP 和 8 位 BMP 更加常见一些。24 位 BMP 一个像素占用 24 位，总共可以有 224 种颜色。24 位色 BMP 无调色板，直接用 3 个字节表示红色、绿色、蓝色分量。而 8 位色 BMP 一个像素占用 8 位，总共可以有 28 种颜色，比 24 位色要少很多。8 位色 BMP 中有调色板，调色板中每一项分别定义了所使用的每种颜色的红绿蓝数值。8 位 BMP 中的颜色值用调色板中索引号表示。24 位色 BMP 和 8 位色 BMP 相比，可以表示的颜色数更多，但相应的数据量也更大，同样大小的 8 位色 BMP 文件大约只有 24 位色 BMP 文件大小的三分之一。

因此，将 24 位色 BMP 转换为 8 位色 BMP 是个很有意义的问题，但同时这也是个比较复杂的问题。这是一个有损变换过程，生成的图像肯定和原始图像有所不同。从 224 种颜色变换为 28 种颜色要经过减色处理，如何精选 256 种颜色并生成调色板是需要解决的一个关键问题。

对于 8 位色 BMP，调色板中没有的颜色需要用调色板中已有的颜色近似处理。这样，原始颜色和近似颜色之间就存在一个误差。如果对误差不进行处理，生成的 8 位色 BMP 颜色就会有较大的失真，这在视觉效果上就比较差一些。因此对颜色误差进行处理是第二个需要解决的关键问题。

8 位色 BMP 支持 RLE8 压缩，如果进行压缩，则文件数据量可能会更小一些。对 8 位色 BMP 进行压缩处理也是另一个比较关键的问题。

## 2 BMP 文件结构

BMP 文件是一种很基本的图像文件。文件格式如表 1 所示。

表 1 BMP 文件格式

| | 字段 | 大小 | 说明 |
|---|---|---|---|
| 文件头 | id | 2 字节 | BMP 文件标识，值为 'BM' |
| | filesize | 4 字节 | 整个 BMP 文件的大小 |
| | reserved | 4 字节 | 保留，值为 0 |
| | dataoffset | 4 字节 | 表示位图数据距离文件头的字节数 |
| 信息头 | headsize | 4 字节 | 信息头长度，值设为 40 |
| | width | 4 字节 | 位图宽度，以像素为单位 |
| | height | 4 字节 | 位图宽度，以像素为单位 |
| | planes | 2 字节 | 位平面数，值设为 1 |
| | bits | 2 字节 | 每个像素占用的位数 |
| | compress | 4 字节 | 压缩方式 0-不压缩 1-RLE8 压缩 |
| | datasize | 4 字节 | 位图数据大小 |
| | hresolution | 4 字节 | 水平分辨率 |
| | vresolution | 4 字节 | 垂直分辨率 |
| | colorused | 4 字节 | 使用的颜色数 |
| | colorimportant | 4 字节 | 重要的颜色数 |
| [调色板] | rgbblue | 1 字节 | 蓝色分量 |
| | rgbgreen | 1 字节 | 绿色分量 |
| | rgbred | 1 字节 | 红色分量 |
| | rgbreserved | 1 字节 | 保留，值为 0 |

其中，调色板是可选的，只存在于 8 位色 BMP 中，调色板中每种颜色用 4 个字节表示，调色板中颜色总数可从 colorused 字段得到或通过计算得到。

然后，开始存储实际的位图数据。对于 24 位 BMP，每个像素用 3 个字节表示，分别对应于蓝、绿、红 3 个颜色分量。对于 8 位 BMP，每个像素用一个字节表示，对应于调色板中的索引号。

BMP 文件按照从下至上、从左至右的顺序依次存储每个像

素。另外，BMP 文件要求每行字节数应该是 4 的整数倍，如果不是，要在行的末尾增加几字节 0，直到每行字节数符合要求。

## 3 8 位色 BMP 调色板的生成

24 位色 BMP 颜色数众多，需要从中精选 256 种能够较好地表示整幅图像的颜色。本程序使用比较常用的流行色算法选取颜色。该算法的基本思想是对整幅图像进行扫描，对每种颜色记录一下出现的次数，再对所有颜色按出现次数从大到小进行排序，选取出现最多的 256 种颜色作为调色板。

本程序使用一个颜色数组表示颜色：

```
struct colorarray{  //颜色数组
  char b;  //蓝色
  char g;  //绿色
  char r;  //红色
  unsigned long count;  //使用次数
};
```

使用如下代码选取颜色：

```
for(i=1;i<=height;i++)  //将 24 位色 BMP 中所有颜色加入颜色
//数组
{
   memcpy(buf,bmp24buf+pos,bytesperline1);
   for(j=1;j<=width*3;j+=3)
   {
    found=0;
      for(k=1;k<=colorcount;k++)
      {
if(abs(colorbuf[k-1].b-buf[j-1])+abs(colorbuf[k-1].g-buf[j])+
      abs(colorbuf[k-1].r-buf[j+1])<30)
       {
      colorbuf[k-1].count++;found=1;
       }
   }
    if(found==0)
   {
     colorbuf[colorcount].b=buf[j-1];
     colorbuf[colorcount].g=buf[j];
        colorbuf[colorcount].r=buf[j+1];
        colorbuf[colorcount].count=1;colorcount++;
   }
}
 pos=pos+bytesperline1;
}
qsort(colorbuf,colorcount,sizeof(struct colorarray),cmp);
//按计数值从大到小排序
```

使用如下代码构造调色板：

```
 //构造调色板
 count=0;
 for(i=1;i<=1024;i++) pallete[i-1]=0;
 for(i=1;i<=colorcount;i++)
 {
    found=0;
  for(j=1;j<=1024;j+=4)
    {
     if(abs(colorbuf[i-1].b-pallete[j-1])+abs(colorbuf[i-1].g-
pallete[j])+
      abs(colorbuf[i-1].r-pallete[j+1])<30)
      {
     found=1;break;
      }
    }
     if(found==0)
     {
    pallete[count]=colorbuf[i-1].b;pallete[count+1]=colorbuf
[i-1].g;
       pallete [count+2]=colorbuf [i-1].r;pallete [count+3]=0;
count=count+4;
     }
   if(count==1024) break;
 }
```

以上代码中，并没有对每一种颜色进行计数，如果那样将耗费比较多的时间。而是对相似颜色进行合并处理，这样大大提高了运算效率。

## 4 颜色误差扩散处理

8 位色 BMP 中最多只有 256 种颜色，没有的颜色要使用调色板中最相近的颜色进行匹配。这样实际的颜色值和相匹配的颜色之间就存在一个误差。如果对误差不进行处理，当 24 位色 BMP 中有较多的渐变色时，8 位色 BMP 将会出现很多明显的条纹，这将大大降低视觉效果。

本程序使用 Floyd-Steinberg 算法对误差进行处理。这个算法的思想是顺序读取每个像素，分别计算蓝绿红 3 原色的误差，将颜色误差扩散到周围的像素中，每个像素都经过这样的处理，颜色误差就被分散到整个位图中，这样可以比较好地提高图像显示效果。对于每个颜色分量误差，向右传递误差值的 7/16，向左下传递误差值的 3/16，向下传递误差值的 5/16，向右下传递误差的 1/16。经过这样的处理，每个像素的误差都被周围像素分摊了，这样整幅图像误差就被平均了，最后的误差接近于 0。

颜色误差处理关键代码如下：

```
b=bmp24buf[pos];g=bmp24buf[pos+1];r=bmp24buf[pos+2];
//根据 24 位 BMP 颜色查找调色板匹配索引
index=GetNearestPaletteIndex(hPalette,RGB(r,g,b));
b1=(unsigned char)bmp8buf[54+index*4];
g1=(unsigned char)bmp8buf[54+index*4+1];
r1=(unsigned char)bmp8buf[54+index*4+2];
diffb=b-b1;  //计算 24 位色与 8 位色蓝色误差
diffg=g-g1;  //计算 24 位色与 8 位色绿色误差
diffr=r-r1;  //计算 24 位色与 8 位色红色误差
```

```
//向右扩散蓝色误差
ch=(int)bmp24buf[pos+3]&0xff;ch=ch+diffb*7/16;
if(ch<0)
    ch=0;
else
    if(ch>255) ch=255;
bmp24buf[pos+3]=(unsigned char)ch&0xff;
```

## 5 RLE8 压缩处理

8 位色 BMP 是支持压缩的。8 位色 BMP 的压缩方式是 RLE8。这是一种行程长度压缩算法。该算法的思想是将重复出现的数据用重复次数+重复数值来表示。可见，这种压缩对于有大量重复数据的位图压缩效率较高。对于重复数据不是很多的位图，往往达不到好的效果。

RLE8 算法使用 2 个字节表示一个像素。第一个字节表示重复次数，第二个字节表示重复数据。可见，使用 RLE8，每次可以压缩最多 255 个重复像素。对于多于 255 个的重复字节，需要分段处理。如果第一个字节为 0，则有以下 4 种情况：

0 0 表示行结束，每行数据末尾有此标记。

0 1 表示文件结束，文件末尾要有此标记。

0 2 x y 表示像素位置改变，当前像素向右移 x 个像素，向下移y 个像素。

0 n … 其中 3<=n<=255，表示 n 个不重复的像素，如果 n 为奇数，要在末尾增加 1 字节 0。

RLE8 压缩代码如下：

```
ch=prevbuf[startpos]; //取第一个字节
if(prevbuf[startpos+1]==ch) //重复字节
{
    count=0;
   for(i=startpos+1;i<=prevbuflen;i++)
   {
     if(prevbuf[i]! =ch) break; //遇到不重复字节退出
     if(count==254) break; //达到最大计数值退出
     count++;
   }
   postbuf[0]=count+1;postbuf[1]=ch;
   *lastpos=i;*postbuflen=2;
   //最末字节为 0 且到达缓冲区尾
   if(prevbuf[prevbuflen-1]==0&&*lastpos>prevbuflen)
 {
      postbuf  [0] =count;postbuf  [2] =0;postbuf  [3] =0;
*postbuflen=4;
    }
    if(prevbuflen-(*lastpos)<=1)
    {
     if(prevbuflen-(*lastpos)==0) //到达末尾
     {
         postbuf[2]=0;postbuf[3]=0;
     *lastpos=prevbuflen;*postbuflen=4;
}
else
{
  if(prevbuflen-(*lastpos)==1) //到达最后一个字节
    {
      postbuf[2]=1;postbuf[3]=prevbuf[prevbuflen-1];
      postbuf[4]=0;postbuf[5]=0;
      *lastpos=prevbuflen;*postbuflen=6;
    }
  }
 }
}
else //不重复字节
{
   len=0;count=0;
   for(i=startpos+1;i<=prevbuflen;i++)
   {
     if(prevbuf[i]==ch) break; //遇到重复字节退出
     if(count==255) break; //达到最大计数值退出
     count++;len++;ch=prevbuf[i];
   }
   if(count==1) //一个不重复字节
{
   postbuf[0]=1;postbuf[1]=prevbuf[startpos];
    *lastpos=i-1;*postbuflen=2;
  }
  if(count==2) //二个不重复字节
{
    postbuf[0]=1;postbuf[1]=prevbuf[startpos];
    postbuf[2]=1;postbuf[3]=prevbuf[i-2];
    *lastpos=i-1;*postbuflen=4;
  }
  if(count>=3) //三个及三个以上不重复字节
{
    postbuf[0]=0;postbuf[1]=count;
 for(j=1;j<=count;j++) postbuf[j+1]=prevbuf[startpos+j-1];
    if(count%2==1) //计数值为奇数
    {
        postbuf[j+1]=0;len++;
    }
    *lastpos=i-1;*postbuflen=2+len;
  }
    }
    if(prevbuflen-(*lastpos)<=1)
  {
     if(prevbuflen-(*lastpos)==0) //到达末尾
     {
   postbuf[(*postbuflen)]=0;postbuf[(*postbuflen)+1]=0;
        *lastpos=prevbuflen;*postbuflen=2+(*postbuflen);
     }
   else
```

（下转第 91 页）

# Flash as3 中实现动态图形的选中与编辑

程海生

**摘　要**：在 Flash 中实现动态图形的选中与编辑，可以使 Flash 动态绘图的交互性更强，可以制作出实用的绘图程序。

**关键词**：教育技术；Flash 应用；动态绘图；选中；编辑

## 1　引言

在教学中使用动态绘图有着静态绘图无法比拟的优点，然而在 Flash 中如何使这些用程序绘制的图形可以用鼠标选中并再次编辑呢？解决这一问题可使其交互性进一步提高。现以绘制函数图像的具体实例说一说在 Flash 中如何实现动态图形的选中与编辑。

实例最终效果：制作一个可以精确绘制数学函数图像的 Flash，可以任意添加函数图像，用鼠标单击其中一个函数图像时可选中相应的图像，并且可以对其进行删除或再编辑等操作。

## 2　设计思路

(1) 创建基类为“MovieClip”的类，在类中定义存储函数参数的变量并编写画函数图像的函数。

(2) 每添加一个函数图像创建一个相应类的对象，当改变函数的参数信息时，由对象中的绘图函数绘制出函数图像，同时将参数信息保存到对象中的相应变量里。

(3) 创建对象的同时为其添加需要的鼠标侦听事件。

(4) 当鼠标单击这些动态图形时，通过重绘不同颜色的函数图像达到选中的效果，同时将对象中变量保存的函数参数信息返还给相应的组件，这样就可以对这些动态图形再次进行编辑了。

## 3　实现过程

### 3.1　准备工作

首先启动 Adobe Flash cs4 创建一个新的 Flash ActionScript3.0 文件，使用默认的舞台大小 550×400，在用程序绘制动态图像前，先用 Flash 绘图工具制作界面和一些静态的图形，这样可以简化程序，当然在制作的过程中也要使用一些技巧才能使这些静态图形快速绘制完成并与程序绘制图形准确的结合。

(1) 绘制坐标系和网格线

以窗口坐标系上的点（275，200）为坐标原点绘制直角坐标系，并绘制上网格，如图 1 所示，此步使用属性面板和对齐面板可轻松完成，注意每个两个刻度间的长度都是 20 个像素，因为后面动态绘图是以 20 个像素为一个单位长度，当然也可以只画坐标系或都不画，并不影响动态图形选中与编辑效果的实现。

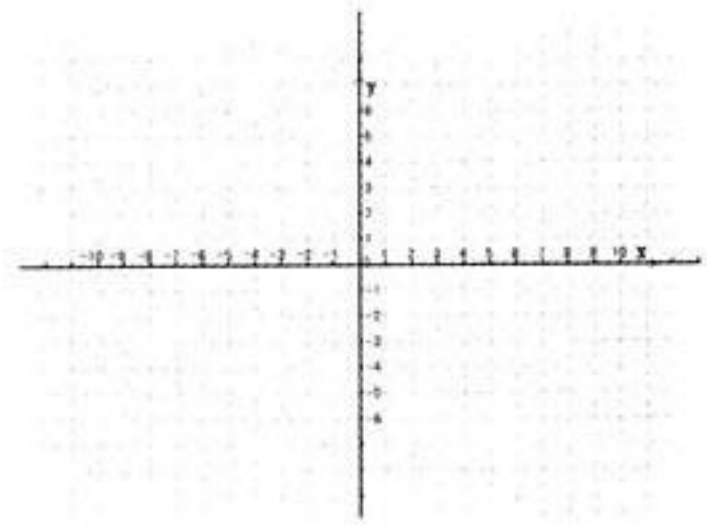


图 1　绘制坐标和网格

(2) 绘制界面

锁定原有图层，在上层添加一个新图层，绘制如图 2 所示的界面，图 2 中间网格部分是下层内容的。单击关键帧选中本层所绘制的界面，按 F8 将其转换为影片剪辑元件，名称为“界面”，再按“del”键将其删除。

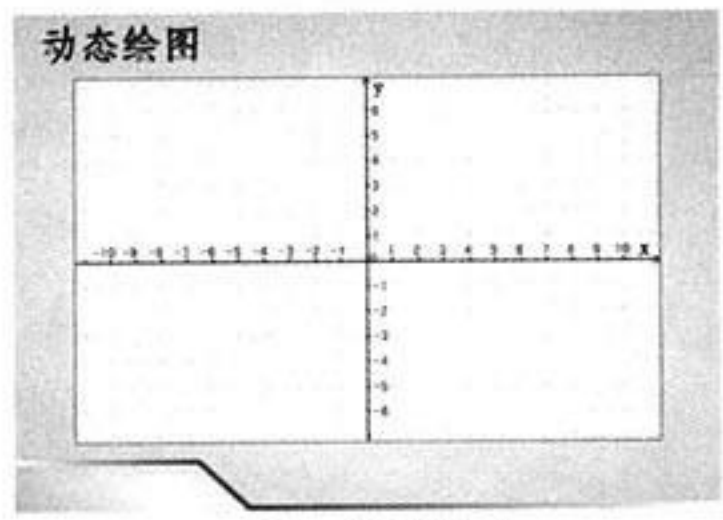



图 2　绘制界面

(3) 在“界面”元件中添加内容

插入一个影片剪辑元件，名称为“抛物线方程”，用文字工具输入文字，插入 3 个 NumericStepper 组件，实例名称分别为：a1，b1，c1。

在属性面板中对 3 个组件实例的参数都设置如下：maxmum：10；minmum：−10；stepSize：0.1；value：0。

打开“界面”元件，将“抛物线方程”元件拖入其中，摆

放到合适的位置，实例名称为：pwxfc。

在适当的位置插入3个“Button”组件，在属性面板参数中将其“Label”值分别设为“添加抛物线”、“确定”和“删除”，实例名分别为：add_pwx、ok和shanchu。如图3所示。


图3 在界面中添加内容

（4）打开库面板，右击“界面”元件，在快捷菜单中选择“属性”，在元件属性中设置类为“jm”。

（5）以“动态图形的选中与编辑.fla”为文件名保存。

### 3.2 编写动态绘图程序

（1）新建一个“ActionScript”文件，保存为“pwx_mc.as”，与“动态图形的选中与编辑.fla”放在同一文件夹中。

在其中输入如下内容：

```
package {
    import flash.display.MovieClip;//导入类 MovieClip
        public class pwx_mc extends MovieClip {//定义
//pwx_mc 类,继承自 MovieClip 类
        public const tuxing:String="pwx";//定义常量 tuxing,
//当同时绘制多种函数图像时用于区分。
  public var n:uint;//用于保存函数图像在 flash 文件中的标号
        public var a1:Number;//这三句的变量用于保存一元
//二次方程(抛物线)的参数
        public var b1:Number;
        public var c1:Number;
        public function pwx_mc() {//在此构造函数未用为空
        }
            public function draw_pwx(color:String) {//定义
//draw_pwx 函数,用于画抛物线图像,参数 color 传递颜色信息。
   //下面的 x0,y0 是以窗口客户区坐标系中点(275,200)为原点,
//一个象素为一个单位长度的平面直角坐标系中的点的坐标 x,y 值。
     var x0:Number=-275;
 //用 x1,y1 表示以窗口客户区坐标系中点(275,200)为原点,20
//个象素为一个单位长度的平面直角坐标系中的点的坐标 x、y值。
            var x1:Number=x0/20;
            //下句用二次函数求出 y1
        var y1:Number=a1*Math.pow(x1,2)+b1*x1+c1;
            var y0:Number=y1*20;
        //x2,y2 为窗口客户区坐标系中点的坐标 x,y 值
            var x2:Number=x0+275;
            var y2:Number = -(y0-200);
            this.graphics.clear();//清除原有图像
    //以下几句通过 color 参数的不同,定义不同的线条样式
            if (color=="blue") {
                this.graphics.lineStyle(3,0x1000FF,0.8);
            } else if (color == "red") {
                this.graphics.lineStyle(3,0xff0000,0.8);
            } else if (color == "yellow") {
                this.graphics.lineStyle(3,0xffcc00,0.8);
            }
            this.graphics.moveTo(x2, y2);//设置画线起点
    //以下使用循环语句绘图,由多条小线段组成函数图像,
    //由于线段都很短,看起来函数图像还是很平滑的
                for (x0=-274; x0<=275; x0++) {
                    x1=x0/20;
                    y1=a1*Math.pow(x1,2)+b1*x1+c1;
                    y0=y1*20;
                    x2=x0+275;
                    y2 = -(y0-200);
                    this.graphics.lineTo(x2, y2);//画线
                    this.graphics.moveTo(x2, y2);
                }
            }
        }
}
```

（2）打开“动态绘形的选中与编辑.fla”在其第一帧上输入语句如下：

```
fscommand("allowscale", "true");//允许缩放
var huitu:MovieClip =new MovieClip();//创建一个 MovieClip
//类对象 huitu,绘图在这个对象中
addChild(huitu);//显示对象 huitu
var jm1:jm =new jm();//创建 jm 类的对象 jm1,这样可以使我们
//静态绘制的"界面"在动态图形的上层
addChild(jm1);
var editing:uint=0;//保存正在编辑的动态图形的标号
var maxn:uint=1000;//以下几句定义一维数组并赋初始值,数组
//用于正在显示的所有动态图象的标号
var mysz = new Array();
for (var i = 0; i<maxn; i++) {
    mysz[i]=0;
}
jm1.pwxfc.visible=false;//不显示抛物线方程
//如果有其他方程也在此设置,如:jm1.yuanfc.visible=false;
        jm1.add_pwx.addEventListener (MouseEvent.CLICK,
addpwx1);//为"添加抛物线"按钮添加鼠标侦听事件
function addpwx1(e:MouseEvent) {
    jm1.pwxfc.visible=true;//显示抛物线方程
//如果有其他方程设为不显示如:jm1.yuanfc.visible=false;
    if (editing! =0) {//如果有正在编辑的图象重设为不编辑状
//态,变为"蓝色"
        if (huitu["f"+editing].tuxing=="pwx") {//如果正在编辑
//的图像是抛物线,重绘抛物线为蓝色
            huitu["f"+editing].draw_pwx("blue");
        }//如果正在编辑的是其他方程的图像写后面如: else
// if (huitu["f"+editing].tuxing=="yuan") { huitu["f"+editing].
```

```
//draw_yuan("blue"); }
            editing=0;//为零表示没有正在编辑的图像
        }
        jm1.pwxfc.a1.value=0;//将抛物线方程中的三个组件清零
        jm1.pwxfc.b1.value=0;
        jm1.pwxfc.c1.value=0;
        editing=1;//以下几句找到一个没有用过的标号
        while (mysz[editing] ! = 0) {
            editing++;
        }
        huitu["f"+editing]=new pwx_mc();//在 huitu 对象中,以""f"
//+editing"为对象命创建一个基于 pwx_mc 的对象
        huitu["f"+editing].n=editing;//将新标号保存到数组中
        huitu.addChild(huitu["f"+editing]);//在 huitu 对象内显示对
//象"f"+editing
        huitu ["f" +editing].addEventListener (MouseEvent.
MOUSE_DOWN,f_pwx);//为对象"f"+editing 添加鼠标按下事
//件,用于选中这个图象。
        function f_pwx(e:MouseEvent) {
        if (editing! =0) {//如果有正在编辑的图像改为不编辑状态
                    if (huitu["f"+editing].tuxing=="pwx") {//正在编
//辑的图像是抛物线,重绘抛物线图像为蓝色
                        huitu["f"+editing].draw_pwx("blue");
                    } //如果是其它图像可以写在后面
            }
            e.target.draw_pwx("red");//将鼠标按下的抛物线图
//像变为红色,即编辑状态
            jm1.pwxfc.visible=true;//显示抛物线方程
        //如果有其它方程都不显示如:jm1.yuanfc.visible=false;
            jm1.pwxfc.a1.value=e.target.a1;//以下三句将一元
//二次方程的三个参数返还给相应组件用于编辑
            jm1.pwxfc.b1.value=e.target.b1;
            jm1.pwxfc.c1.value=e.target.c1;
      editing=e.target.n;//将正在编辑图像的标号设为这个图像的
        }
        huitu ["f" +editing].addEventListener (MouseEvent.
MOUSE_OVER,f_pwx2); //为对象"f"+editing 添加 MOUSE_O
//VER 事件当鼠标指向这个函数图象时变为黄色
        function f_pwx2(e:MouseEvent) {
            e.target.draw_pwx("yellow");
        }
        huitu ["f" +editing].addEventListener (MouseEvent.
MOUSE_OUT,f_pwx3); //为对象"f"+editing 添加MOUSE_OUT
//事件当鼠标离开这个函数图象时变为原有颜色
        function f_pwx3(e:MouseEvent) {
            if (e.target.n==editing) {
                e.target.draw_pwx("red");
            } else {
                e.target.draw_pwx("blue");
            }
        }
}
//下面几句分别为方程中三个 NumericStepper 组件添加监听事件
jm1.pwxfc.a1.addEventListener(Event.CHANGE,pwx);
jm1.pwxfc.b1.addEventListener(Event.CHANGE,pwx);
jm1.pwxfc.c1.addEventListener(Event.CHANGE,pwx);
function pwx(e:Event) {
huitu["f"+editing].a1=jm1.pwxfc.a1.value;//以下三句将方程中
//的三个 NumericStepper 组件的值保存到对象的相应变量中
        huitu["f"+editing].b1=jm1.pwxfc.b1.value;
        huitu["f"+editing].c1=jm1.pwxfc.c1.value;
        huitu["f"+editing].draw_pwx("red");//在对象中绘制红色一
//元二次方程图像
        mysz[editing]=editing;//保存函数标号
}
jm1.shanchu.addEventListener(MouseEvent.CLICK,shanchu1);
//为"删除"按钮添加鼠标单击事件
function shanchu1(e:MouseEvent) {
        jm1.pwxfc.visible=false;//不显示抛物线方程
//如有其他方程也设为不显示如:jm1.yuanfc.visible=false;
if (editing! =0) {//有正有编辑的图像,移除这个图像所在的对象
            huitu.removeChild(huitu["f"+editing]);
            mysz[editing]=0;//将数组中相应元素清零
            editing=0;// editing=0 表示没有正在编辑的图象
        }
}
jm1.ok.addEventListener(MouseEvent.CLICK,ok1);//为"确定"
//按钮添加鼠标单击事件
function ok1(e:MouseEvent) {
        jm1.pwxfc.visible=false;//不显示抛物线方程
        //如果有其他方程也在此设为不显示
        if (editing! =0) //如果有正有编辑的图像,将其变为不编辑
//状态,其蓝色
            if (huitu["f"+editing].tuxing=="pwx") {//如果正在编辑
//的图像是抛物线变为不编辑状态"蓝色"
                huitu["f"+editing].draw_pwx("blue");
            } //如果正在编辑的图像是其他函数图像也在后面
            editing=0;// editing=0;表示没有正在编辑的图像
        }
}
stop();
```

至此 Flash 动态绘图程序已经有了选中和编辑的功能了，可以测试一下看看效果如图 4 所示。

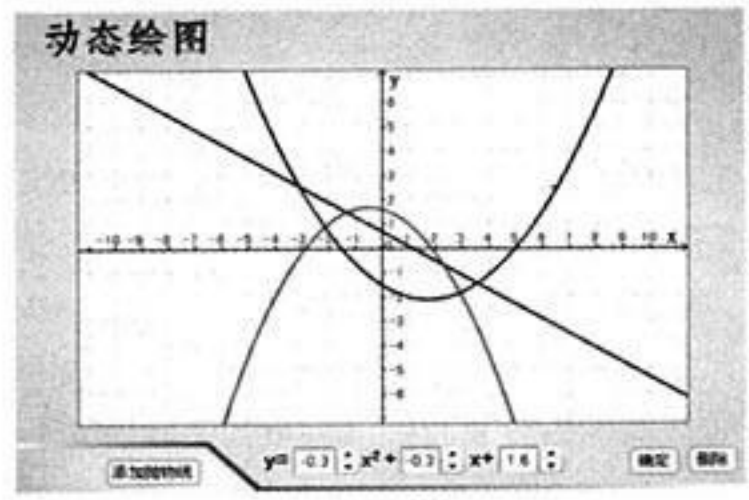



图 4　运行效果

（收稿日期：2012-03-20）

# 基于Java的“连连看”游戏

仇 宾

**摘 要**：“连连看”是一款风靡网络的小游戏，版本也非常多，使用Java语言实现了一个单机版的连连看游戏，以期读者能通过游戏的制作来学习Java语言，达到事半功倍的效果。

**关键词**：Java语言；游戏编程；连连看

## 1 引言

“连连看”游戏在网上种类非常多，比如“水果连连看”、“宠物连连看”等，虽然版本各种各样，但是其基本玩法，或者说基本算法是相同的，就是显示一些图标，让用户依次去点击两个图标，如果这两个图标相同，并且这两个图标通过直线可以相连，或者通过直角相连，或者通过双折线相连就可以消掉，消掉所有图标即为胜利。如图1所示。

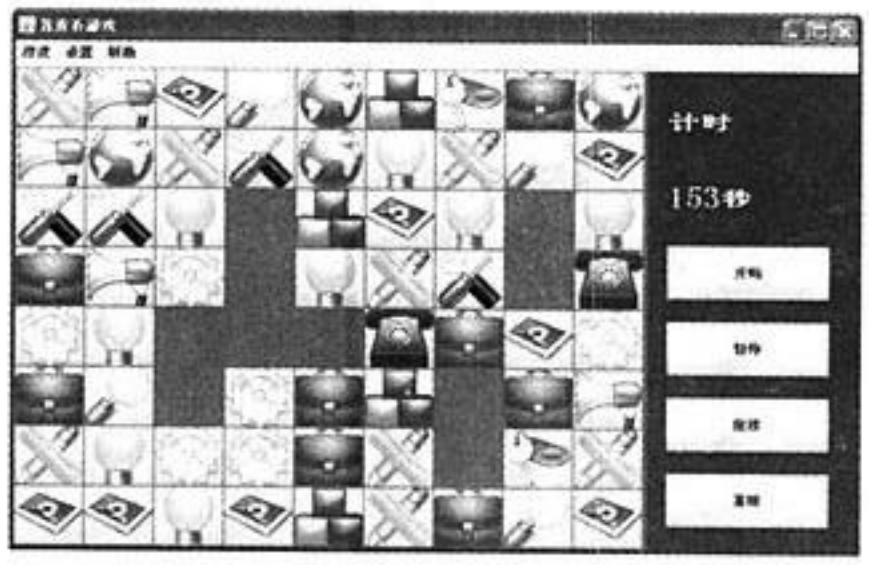



图1 正在进行中的连连看游戏

通过该游戏的制作，可以对Java的基础语法、Java图形界面以及简单的算法设计有一个比较全面的了解。

## 2 设计要求

为了避免叙述的繁琐，游戏设计要求如下：

(1) 制作如图1所示的游戏界面，尽量做到美观大方，使用方便。

(2) 当两个图标相同，且通过直线相连、直角相连、双折线相连时，能够消掉图标。

(3) 能够让游戏随时暂停，然后继续。

(4) 当游戏进行到一定程度，无法消除剩余图标时，要能够提供重排功能对图标进行重排，从而让游戏继续进行。

## 3 实现思路

介绍程序中的几个难点及实现思路，具体的代码在后文给出。

### 3.1 界面设计

需要实现如图2所示的界面。

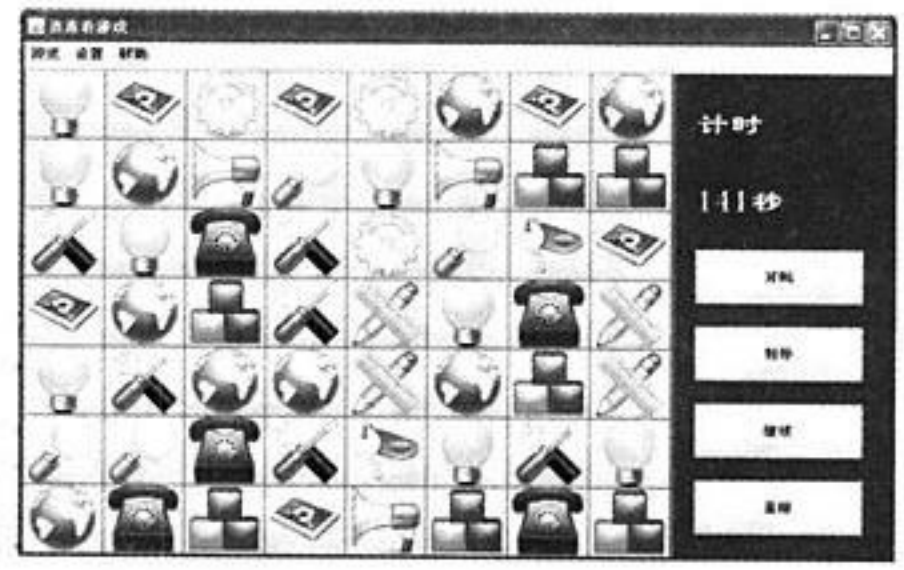



图2 连连游戏界面

整个界面分为3个区域：菜单区、功能区、游戏区。首先在窗体上放置3个面板JPanel，分别存放3个区域，如图3所示。


图3 界面的设计

系统菜单区放置菜单即可，用户游戏区放置一个8*9的按钮数组来构成游戏界面，用户交互区放置开始、暂停等功能按钮以及提示信息。

### 3.2 生成游戏区

运行程序后，游戏区并不显示按钮数组，当点击开始按钮后，再自动生成。在生成按钮时要求按钮上的图案是随机的，且每个图案必须是偶数，否则会出现无法消除的按钮。如何实

现呢？可以这样考虑：假设有 12 个图片，把图片名字按照数字序号从 0 到 11 命名；假设游戏区共 72 个按钮，那么产生 36 个 12 以内的随机数字（每个随机数字代表一个图片），放入一个 ArrayList 中，最后使用 ArrayList 的 addAll 方法对已经产生的 36 个随机数字复制一份，这样就获得了 72 个随机数字，并且是成对的。因为一个数字对应一个图片，所以 72 个按钮需要的图片就生成了。代码如下：

```
Random random = new Random();
int imagenum = ROW * COL;
for(int i=0; i<imagenum/2; i++){
    imageIndex.add(random.nextInt(12)+"");// 生成随机数索
//引放入集合
}
imageIndex.addAll(imageIndex);// 连接集合使每个索引都是
//成偶数的
```

### 3.3 联通消除

点击相连且图片相同的两个按钮可以把两个按钮消除掉。相连有 3 种情况：直线相连、直角相连、双折线相连。

要消掉相同图标的按钮，首先要想办法判断两个按钮的图标是否相同，这里提示大家一个方法：按钮有个 setActionCommand 方法和 getActionCommand，把图标名字设置为一个按钮的 ActionCommand，然后通过对比两个按钮的 ActionCommand 就可以知道是否具有相同的图标。

消除按钮就要判断两个按钮是否连通，连通有 3 种情况，对应3 种算法，下面详细说明。算法判断两个点是否连通，那如何跟两个按钮建立联系呢？按钮的位置就可以视作一个点（Java 中用 Point 类表示一个点）。这样用一个点代表按钮位置，用 ActionCommand 代表按钮上的图标。就很容易操作了。下面先说清楚实现原理，具体代码一会再给出。

（1）直线相连：只要两点之间横坐标或纵坐标相同，即表示两点在同行或同列；然后判断两点之间有没有障碍，无则联通，如图 4 所示。

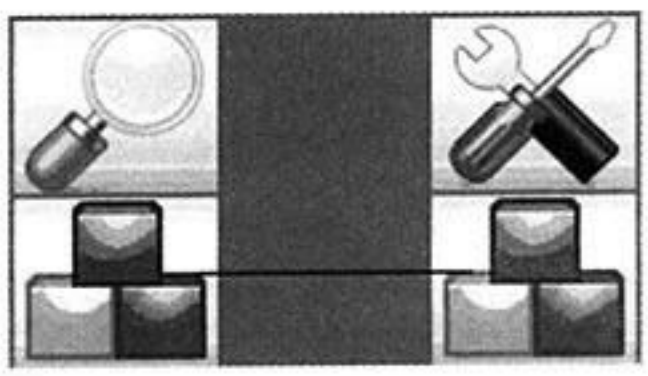


图 4　直线相连

（2）直角相连：两个点的坐标可以生成一个新的点，如果两点都可以与该新点联通，则说明联通。注意：新点有两个，如图 5 所示一个在空白区域，另外喇叭图标处也可以生成一个，但是因为喇叭图标处不是空点，因此不用该点。但在实际中，新点的两种可能都要考虑到，哪个能用、哪个不能用。

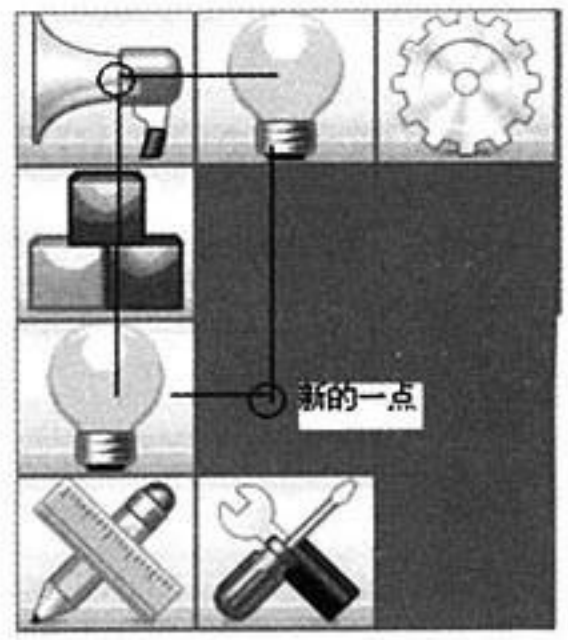



图 5　直角相连

（3）双折线相连：在单击的第一点处，向 4 个方向搜索有无空点，如果有则生成一个新点，如果该新点与另一个按钮联通（直角联通），则说明联通。如果该新点不能与按钮连通，则继续向该新点的方向前进一个位置，再次判断该位置是否为空点，是否可以和另一个按钮连通，以此类推，如图 6 所示。

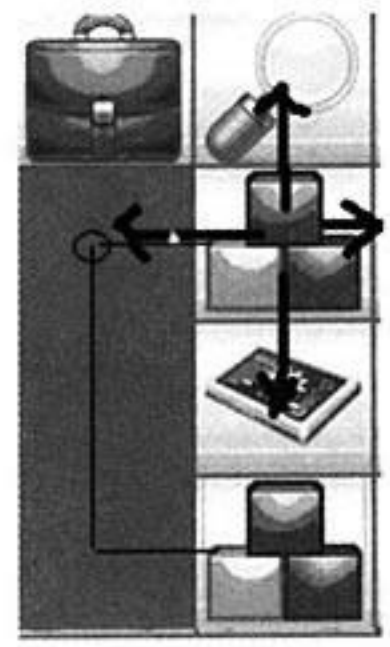


图 6　双折线相连

### 3.4 重排

当游戏进行到一定程度，剩余的按钮无法消掉，这时就需要对按钮位置进行重新排列，从而让游戏继续进行。解决这一问题的思路如下：首先对所有按钮进行搜索，把剩余按钮的 ActionCommand（ActionCommand 和该按钮显示的图片序号一致，如果按钮被消掉则 ActionCommand 为空）放入一个集合内；然后依次搜索剩余按钮，找到一个剩余按钮后，保持按钮的位置不动，从集合中随机取出一个 ActionCommand 赋给该按钮，这样就完成了对剩余按钮的重排。

## 4 具体实现

打开 Eclipse，新建一个类 LinkGame，该类继承自 JFrame，并且需要实现 ActionListener 方法。

```
public class LinkGame extends JFrame implements
ActionListener{
}
```

在该类中添加如下变量和方法，如表 1 和表 2 所示。

表 1 LinkGame 类中的成员变量

| 变量类型 | 变量名 | 说明 |
|---|---|---|
| final static int | ROW | 游戏区 8 行 9 列共 72 个按钮 |
| final static int | COL | |
| JPanel | functionPanel | 功能区面板 |
| JPanel | gamePanel | 游戏区面板 |
| JButton[][] | dots | 按钮数组 |
| JLabel | timestr | 内容是“剩余时间” |
| JLabel | timecount | 计时 |
| Timer | timer | |
| JButton | start | 开始按钮 |
| JButton | pause | 暂停 |
| JButton | conti | 继续 |
| JButton | hint | 重排 |
| static int | count | 记录鼠标单击次数 |
| static int | time | 记录游戏剩余时间 |
| Point | p1, p2 | 保存两个被单击按钮的坐标 |
| LinkedList<Point> | list | 按钮坐标集合 |
| ArrayList<String> | imageIndex | 存放图像索引 |
| LinkedList<String> | linklist | 重排时存放按钮 |

表 2 LinkGame 类中的方法

| 方法名 | 方法功能 |
|---|---|
| public void addmyMenu () | 添加菜单 |
| public void addFunctionPanel () | 添加功能区 |
| public void addGamePanel () | 添加游戏区 |
| private class ButtonEvents implements ActionListener | 游戏区按钮被单击 |
| public boolean lineCheck (Point p1, Point p2) | 是否直线联通 |
| public boolean secendLine (Point p1, Point p2) | 是否直角联通 |
| public boolean triLine (Point p1, Point p2) | 是否双折线联通 |
| public void reSet () | 当剩余按钮无法消除时进行重排 |
| public void actionPerformed (ActionEvent e) | 功能区按钮被单击时 |

上述方法的具体实现如下：

## 4.1 构造方法

```
// 构造方法建立游戏界面
    public LinkGame() {
        this.setSize(780, 500);
        this.setTitle("连连看游戏");
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     addmyMenu(); // 添加菜单
        addFunctionPanel(); // 添加功能区
        this.setVisible(true);
    }
```

## 4.2 菜单

菜单的功能均没有实现，读者可以自行添加：

```
// 添加菜单
    public void addmyMenu() {
        JMenuBar menuBar = new JMenuBar();
        this.setJMenuBar(menuBar);
        JMenu menuGame = new JMenu("游戏");
        JMenu menuSet = new JMenu("设置");
        JMenu menuHelp = new JMenu("帮助");
        menuBar.add(menuGame);
        menuBar.add(menuSet);
        menuBar.add(menuHelp);
        menuGame.add(new JMenuItem("打开"));
        menuGame.add(new JMenuItem("保存"));
        menuGame.add(new JMenuItem("退出"));
    menuSet.add(new JCheckBoxMenuItem("音乐开关"));
        JMenu choice = new JMenu("难度选择");
        menuSet.add(choice);
        menuSet.add(new JMenuItem("背景色"));
        menuSet.add(new JMenuItem("提示"));
        menuHelp.add (new JMenuItem (" 关 于 ", new
ImageIcon("image/2.gif")));
        menuHelp.addSeparator();
        menuHelp.add(new JMenuItem("帮助"));
        ButtonGroup group = new ButtonGroup();
                JRadioButtonMenuItem rbm1 = new
JRadioButtonMenuItem("初级难度");
                JRadioButtonMenuItem rbm2 = new
JRadioButtonMenuItem("中级难度");
                JRadioButtonMenuItem rbm3 = new
JRadioButtonMenuItem("高级难度");
        group.add(rbm1);
        group.add(rbm2);
        group.add(rbm3);
        choice.add(rbm1);
        choice.add(rbm2);
        choice.add(rbm3);
    }
```

## 4.3 功能区和游戏区的界面

```
// 添加功能区
    public void addFunctionPanel() {
        start.setPreferredSize(new Dimension(150, 50));
        pause.setPreferredSize(new Dimension(150, 50));
        conti.setPreferredSize(new Dimension(150, 50));
```

```
        hint.setPreferredSize(new Dimension(150, 50));
    timecount.setPreferredSize(new Dimension(150, 50));
        timestr.setPreferredSize(new Dimension(150,50));
        timestr.setFont(new Font("隶书",Font.BOLD,28));
    timecount.setFont(new Font("隶书",Font.BOLD,28));
        timecount.setForeground(Color.yellow);
        timestr.setForeground(Color.yellow);
        start.addActionListener(this);
        pause.addActionListener(this);
        conti.addActionListener(this);
        hint.addActionListener(this);
  functionPanel.setPreferredSize(new Dimension(200, 500));
        functionPanel.setBackground(Color.BLUE);
        functionPanel.setLayout(new FlowLayout(FlowLay
out.CENTER, 5, 20));
        functionPanel.add(timestr);
        functionPanel.add(timecount);
        functionPanel.add(start);
        functionPanel.add(pause);
        functionPanel.add(conti);
        functionPanel.add(hint);
        this.add(functionPanel, BorderLayout.EAST);
    }
    // 添加游戏区
    public void addGamePanel() {
    gamePanel.setPreferredSize(new Dimension(580, 500));
    gamePanel.setLayout(new GridLayout(ROW, COL));
        gamePanel.setBackground(Color.RED);
        Random random = new Random();
        int imagenum = ROW * COL;
        for(int i=0; i<imagenum/2; i++){
            imageIndex.add(random.nextInt(12)+"");// 生成
//随机数索引放入集合
        }
         imageIndex.addAll(imageIndex);// 连接集合使每个
//索引都是成偶数的
        int num=0;      //记录从集合中取出的图标做引数
        for (int i = 0; i < ROW; i++) {
            for (int j = 0; j < COL; j++) {
    String newNumber = imageIndex.get(num++);
                    dots [i][j] = new JButton ("", new
ImageIcon("image/" + newNumber
                            + ".png"));
        dots[i][j].addActionListener(new ButtonEvents());
// 事件监听注册
        dots[i][j].setActionCommand(newNumber);
// 设置按钮的 actioncommand 名
                    gamePanel.add(dots[i][j]);
            }
        }
        this.add(gamePanel, BorderLayout.WEST);
    }
```

## 4.4 单击游戏区按钮激发的事件

```
// 事件类,监听按钮单击事件
    private class ButtonEvents implements ActionListener {

        public void actionPerformed(ActionEvent e) {
            count++;

            for (int i = 0; i < ROW; i++) {
                for (int j = 0; j < COL; j++) {
                    if (e.getSource() == dots[i][j]) {
                        Point p = new Point(i, j);
// 单击按钮则把按钮位置封入 p 中
                list.add(p); // 点 p 放入集合 list 中
                    }
                }
            }
        if (count % 2 == 0) { // 如果是偶数次单击按钮
    p1 = list.getFirst(); // 从集合中取出代表两个按钮的点
                p2 = list.getLast();
                // 如果两次点击的是同一个按钮
                if(p1.x == p2.x && p1.y == p2.y){
                    list.clear();
                    return;
                }
                // 这两个点是否可以消掉
                if  (lineCheck  (p1, p2) || secendLine(p1,
p2) || triLine(p1, p2)) {
                    dots[p1.x][p1.y].setVisible(false);
                    dots[p2.x][p2.y].setVisible(false);
            dots[p1.x][p1.y].setActionCommand("");
            dots[p2.x][p2.y].setActionCommand("");
                }
                list.clear(); // 清空集合
                count = 0;
            }
        }
    }
```

## 4.5 判断两个按钮相连的 3 种算法

```
// 是否直线联通
    public boolean lineCheck(Point p1, Point p2) {
        int absDistance = 0; // 两点之间的距离
        int spaceCount = 0; // 两点之间空按钮的数量
        // 如果两个按钮的图标不相同
        if (! dots[p1.x][p1.y].getActionCommand().equals(
                dots[p2.x][p2.y].getActionCommand())) {
            return false;
        }
        if (p1.x == p2.x || p1.y == p2.y) {
            if (p1.x == p2.x && p1.y ! = p2.y) {
                // 绝对距离(中间隔着的空格数)
            absDistance = Math.abs(p1.y – p2.y) – 1;
```

```
                    // 正负值
                    int zf = (p1.y – p2.y) > 0 ? –1 : 1;
                    for (int i = 1; i <= absDistance; i++) {
                        if (dots [p1.x] [p1.y + i * zf].
getActionCommand().equals("")) {
                            // 空格数加 1
                            spaceCount += 1;
                        } else
                    break;// 遇到阻碍就不用再探测了
                    }

                } else if (p1.y == p2.y && p1.x ! = p2.x) {
                absDistance = Math.abs(p1.x – p2.x) – 1;
                    int zf = (p1.x – p2.x) > 0 ? –1 : 1;
                    for (int i = 1; i <= absDistance; i++) {
                        if (dots [p1.x + i * zf] [p1.y].
getActionCommand().equals("")) {
                            spaceCount += 1;
                        } else
                            break;
                    }
                }
                if (spaceCount == absDistance) {
                    // 可联通
                    return true;
                } else {
                    return false;
                }
            } else {
                // 不是同行同列的情况所以直接返回 false;
                return false;
            }
        }
        // 是否直角联通
        public boolean secendLine(Point p1, Point p2) {
        // 第一个直角检查点,如果这里为空则赋予相同值供检查
            Point checkP1 = new Point(p1.x, p2.y);
        // 第二个直角检查点,如果这里为空则赋予相同值供检查
            Point checkP2 = new Point(p2.x, p1.y);
            // 第一个直角点检测
            if (dots[checkP1.x][checkP1.y].getActionCommand
().equals("")) {
                dots [checkP1.x] [checkP1.y].setActionCommand
(dots[p1.x][p1.y]
                        .getActionCommand());
                if (this.lineCheck (p1, checkP1) && this.
lineCheck(checkP1, p2)) {
                dots[checkP1.x][checkP1.y].setActionCommand("");
                    return true;
                } else {
                dots[checkP1.x][checkP1.y].setActionCommand("");
                }
            }
            // 第二个直角点检测
            if (dots[checkP2.x][checkP2.y].getActionCommand
().equals("")) {
                dots [checkP2.x] [checkP2.y].setActionCommand
(dots[p2.x][p2.y]
                        .getActionCommand());
                if (this.lineCheck (p1, checkP2) && this.
lineCheck(checkP2, p2)) {
                dots[checkP2.x][checkP2.y].setActionCommand("");
                    return true;
                } else {
                dots[checkP2.x][checkP2.y].setActionCommand("");
                }
            }
            return false;
        }
        // 是否双折线联通
        public boolean triLine(Point p1, Point p2) {
            int i;
            Point checkP = new Point(p1.x, p1.y);
            // 四向探测开始
            for (i = 4 – 1; i >= 0; i––) {
                checkP.x = p1.x;
                checkP.y = p1.y;
                // 向右
                if (i == 3) {
                    while ((++checkP.x < dots.length)
                && dots[checkP.x][checkP.y].getActionCommand().equals(
                            "")) {
                        dots [checkP.x] [checkP.y].
setActionCommand(dots[p1.x][p1.y]
                                .getActionCommand());
                        if (secendLine(checkP, p2)) {
                    dots[checkP.x][checkP.y].setActionCommand("");
                            return true;
                        } else {
                    dots[checkP.x][checkP.y].setActionCommand("");
                        }
                    }
                }// 向下
                else if (i == 2) {
                    while ((++checkP.y < dots[checkP.x].length)
                            && dots[checkP.x][checkP.y].
getActionCommand().equals(
                                    "")) {
                    dots[checkP.x][checkP.y].setActionCommand(dots[p1.x][p1.y]
                                    .getActionCommand());
                        if (secendLine(checkP, p2)) {
                    dots[checkP.x][checkP.y].setActionCommand("");
                            return true;
                        } else {
```

```
                dots[checkP.x][checkP.y].setActionCommand("");
                            }
                        }
                    }// 向左
                    else if (i == 1) {
                        while ((--checkP.x >= 0)
 && dots[checkP.x][checkP.y].getActionCommand().equals(
                                                "")) {
                                    dots  [checkP.x] [checkP.y].
setActionCommand(dots[p1.x][p1.y]
                                            .getActionCommand());
                                if (secendLine(checkP, p2)) {
                dots[checkP.x][checkP.y].setActionCommand("");
                                    return true;
                                } else {
                dots[checkP.x][checkP.y].setActionCommand("");
                                }
                        }
                    }// 向上
                    else if (i == 0) {

                        while ((--checkP.y >= 0)
                                && dots[checkP.x][checkP.y].
getActionCommand().equals(
                                                "")) {
                                    dots  [checkP.x][checkP.y].
setActionCommand(dots[p1.x][p1.y]
                                            .getActionCommand());
                                if (secendLine(checkP, p2)) {
                dots[checkP.x][checkP.y].setActionCommand("");

                                    return true;
                                } else {
                dots[checkP.x][checkP.y].setActionCommand("");
                                }
                        }
                    }
                }
                // 四个方向寻完都没找到可行的 checkP 点,所以只
//好返回 false
                return false;
        }
```

### 4.6 重排功能

```
// 当剩余按钮无法消除时进行重排
    public void reSet(){
        for(int i=0; i<ROW; i++){
            for(int j=0; j<COL; j++){
        if(!  dots[i][j].getActionCommand().equals("")){
        linklist.add(dots[i][j].getActionCommand());
                }
            }
```

```
            }
            Random rd = new Random();
            for(int i=0; i<ROW; i++){
                for(int j=0; j<COL; j++){
            if(!  dots[i][j].getActionCommand().equals("")){
                            int c = rd.nextInt(linklist.size());
                            dots [i] [j].setIcon (new  ImageIcon ("
image/"+linklist.get(c)+".png"));
                dots[i][j].setActionCommand(linklist.get(c));
                            linklist.remove(c);
                        }
                    }
                }
            }
```

### 4.7 功能区按钮的实现

```
// 功能区按钮被单击时
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == start){
            gamePanel.removeAll();
            addGamePanel(); // 添加游戏区
            reSet();
            timer.start();
        }else if(e.getSource() == pause){
            gamePanel.setVisible(false);
            timer.stop();
        }else if(e.getSource() == conti){
            gamePanel.setVisible(true);
            timer.start();
        }else if(e.getSource() == hint){
            reSet();
        }
        timecount.setText(time--+"秒");
    }
```

最后，加上主方法即可，如下：

```
public static void main(String[] args) {
        new LinkGame();// 匿名对象
    }
```

此时，运行程序就可以进行连连看游戏了。

## 5 结语

通过这个游戏可以比较全面地学习 Java 中的图形界面编程和基本的游戏算法。游戏还有一些功能没有给出，比如用户找不到能消除的按钮时要给出提示，菜单中的音乐控制，难度控制等，有兴趣的读者可以去自行钻研了。

（收稿日期：2012-01-10）

# 基于.NET 的文件备份程序设计

沈建涛

**摘　要**：利用 SYSTEM.IO 类库，设计了一个文件备份程序，用以备份磁盘上大量的文件。

**关键词**：源文件夹；目标文件夹；源文件；目标文件；修改日期时间

在日常的计算机使用过程中，一定会生成很多文件，例如大量的 Word、Excel、下载的网页文件、图片文件等，如果不对这些文件进行备份，一旦磁盘损坏，会造成不可估量的损失。而一般的备份方法通过 Windows 操作系统提供的复制与粘贴的方式进行备份，但这有许多不足之处：首先，每一次备份，不管文件有没有修改过，都需要重新进行一次复制与粘贴，这样每次备份所需花费的时间一定不小于第一次的备份所花费的时间，即要花很长的时间，其次，由于大量的磁盘的读写操作会缩短了磁盘的使用寿命；再次，利用复制与粘贴这种方法，很容易会遗漏源盘上 uWSMEWORK 2.0 DI 隐藏的子文件夹和文件；最后最可怕的是，如果目标盘上的文件经过修改后比源盘上对应的文件要新的话，由于遭到源盘上文件的覆盖，会造成数据的丢失。本程序的设计，利用了.NETFramework 中提供的有关文件及文件夹管理的类，解决了以上这些问题。

## 1　设计原理

对于一个源文件，首先判断目标盘上是否存在该文件，如果不存在，则直接复制过去，否则分别获取源文件与目标文件的修改日期时间的这个属性，通过进行对比新旧，决定是否进行备份。如果源文件的修改日期时间比目标文件的修改日期时间新，则覆盖目标文件，否则不复制。对于源盘上的文件夹，判定目标盘对应的路径下是否存在，如果不存在，则在目标上新建该文件夹。

## 2　文件与文件夹处理

根据源文件夹下文件的类型，对文件与文件夹进行分别处理。

### 2.1　文件处理

（1）针对一个源文件夹，创建一个 DirectoryInfo 类的对象。

（2）通过 DirectoryInfo 的 GetFiles 方法，获取源文件夹下所有的文件对象序列，保存到一个数组中。

（3）通过 FOR EACH 循环语句分别获取数组中的每个文件对象。

（4）针对一个文件对象，查找备份盘上对应的的路径下是否存在该文件（通过 FILE 类的 EXIST 方法），如果不存在，则直接复制过去（通过 FILE 类的 COPY 方法）；如果存在，则通过 FILEINFO 类创建该文件对象，然后比较源文件与目标文件的修改日期时间（通过文件对象 LASTWRITETIME 这个属性），如果源文件对象的修改日期时间比目标文件对象的修改日期时间要新，即源文件的 LASTWRTETIME 大于目标文件的 LASTWRITETIME，则直接复制过去，否则跳过，处理下一个文件对象。

### 2.2　文件夹处理

（1）通过 DirectoryInfo 的 GetDirectories 方法，获取源文件夹下所有的子文件夹对象序列，保存到下一个数组中。

（2）通过 FOR EACH 循环语句分别获取每个子文件夹对象。

（3）针对一个子文件对象，查找备份盘上相同的父文件夹是否存在相同名称的子文件夹（通过 Directory 的 EXIST 方法），如果存在，则对该子文件夹的文件按前面的处理方式进行，如果不存在，则首先在在目标盘对应的父文件夹下创建子文件夹（通过 Directory.CreateDirectory 的方法），再按文件的处理方式进行备份。

## 3　算法流程图

### 3.1　文件备份的算法流程

如图 1 所示。

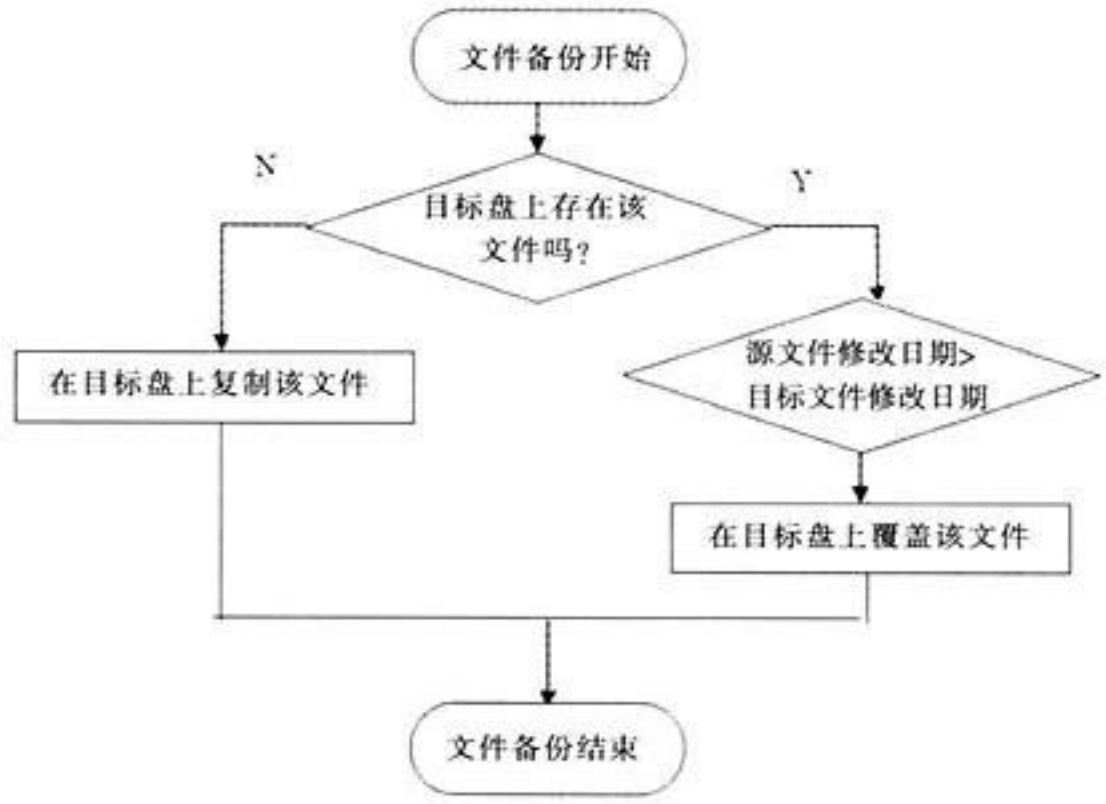



图 1　文件备份算法流程图

### 3.2 文件夹备份的算法流程

如图 2 所示。

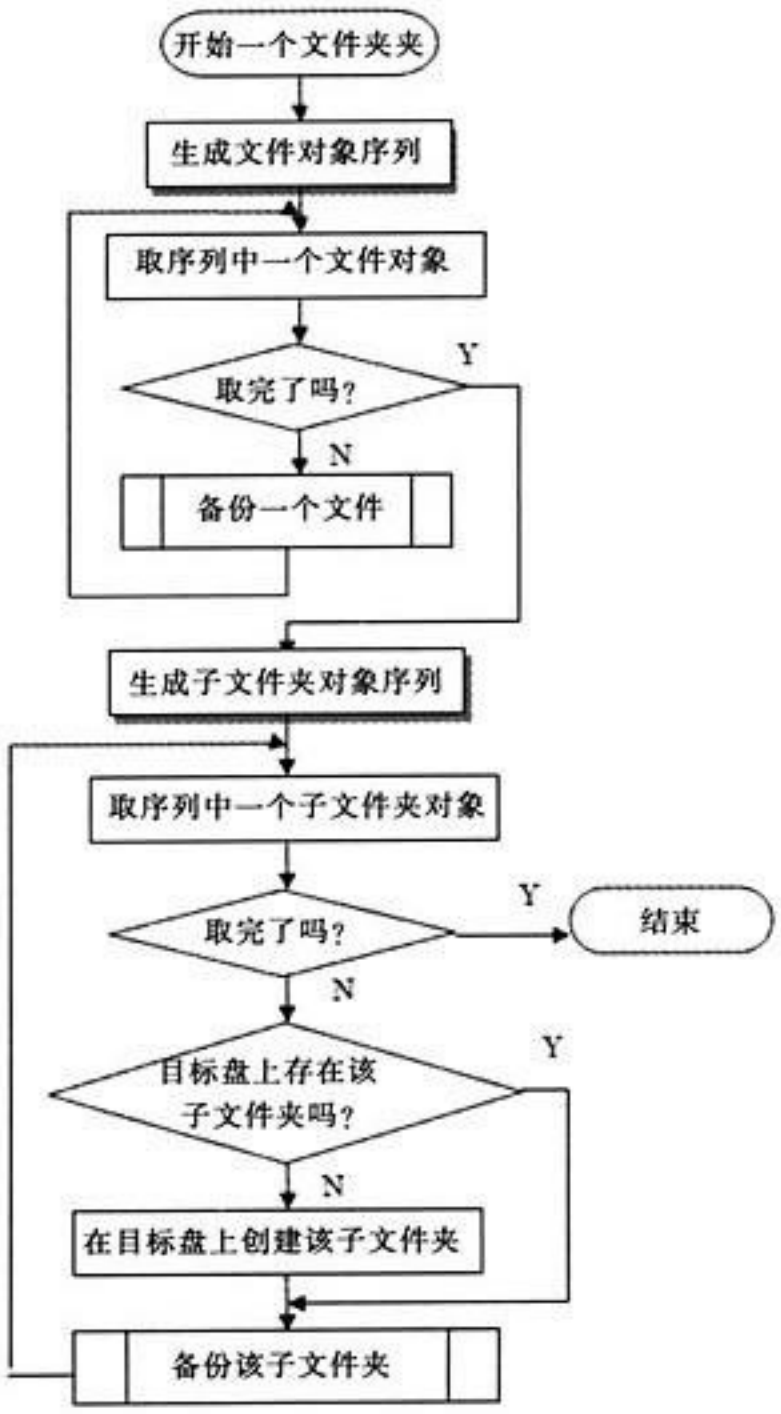



图 2 文件夹备份的算法流程

## 4 关键技术

### 4.1 关键类及其属性、方法

(1) System.IO.File 类及其方法：Exists、Copy、Delete。

(2) System.IO.FileInfo 类及其属性：LastWriteTime。

(3) System.IO.Directory 类及其方法： Exists、CreateDirectory。

(4) System.IO.DirectoryInfo 类及其方法:GetDirectories、GetFiles。

### 4.2 递归技术

建立一个程序过程，把目标文件夹作为该过程的参数，该过程对文件夹下的文件进行直接处理；对子文件夹，首先判定在目标盘上是否存在，如果不存在，先在目标盘上创建该文件夹，然后递归调用过程自身；回归的条件，取完文件夹下子文件夹后进行回归。

## 5 关键代码

基于 Viusal Basic 代码如下：

```
    Dim 文件对象数组 As FileInfo() = 源文件夹.GetFiles()
'获取源文件夹下所有的文件,生成一个文件对象数组
  For Each 文件对象 In 文件对象数组   '循环获取数组中每
'个文件对象
    If Not File.Exists(目标盘.文件对象) Then   '目标文件夹
'是否存在该文件
      File.Copy(源文件, 目标文件)   '不存在,进行直接复制
    Else       '目标文件夹存在该文件
      If fi.LastWriteTime > dfi.LastWriteTime Then '比较
'修改日期时间属性
    File.Delete(旧的目标文件)   '先删除旧的目标文件
        File.Copy(源文件, 目标文件)       '进行复制
        End If
      End If
    Next FOR
    Dim 子文件夹对象数组 As DirectoryInfo () = 源文件夹.
GetDirectories() '获取源文件夹下所有的子文件夹,生成一个文
'件夹对象数组
    For Each 文件夹对象 In 子文件夹对象数组 '循环获取
'数组中每个文件夹对象
      If Not Directory.Exists(目标盘.文件夹对象) Then
'目标盘上是否存在该文件夹
        Directory.CreateDirectory(path2)   '不存在,在
'目标盘上创建该文件夹
      End If
      CALL 对该文件夹的文件进行备份
    Next FOR
```

## 6 注意事项

在程序的设计的过程中，为了操作的方便，还需用到一些控件，如 FolderBrowserDialog 控件等，另外作为使用者，一定要注意计算机的系统日期时间要与当时的实际日期时间一致，否则会造成 LastWriteTime 的不正确，从而造成不能完全备份或旧文件会覆盖新文件。

## 7 结语

(1) 使用技巧：利用本程序可以完成对文件的管理，例如，磁盘 D 上有一个文件夹 DIR1、在 E 盘上有一个文件夹 DIR2，通过 DIR1 备份到 DIR2，再通过 DIR2 备份到 DIR1，这样，两个文件夹下的文件就完全一样了。

(2) 展望：可以通过创建一个数据文件，来保存文件的一些信息以及备份的一些情况，以便更好地对文件进行管理，但要注意，如果文件数量很多，该数据文件也就较大，一定要根据自己的实际情况，有所选择。

(收稿日期：2012-12-12)

# 利用端口转发技术实现局域网穿透（下）

杨 勇

**摘 要**：内网主机上的转发器要处理大量外网客户端的并发连接，因此需要有一种高效并行处理多连接的I/O机制。采用完成端口模型（IOCP）管理并发连接，为提高完成端口的执行性能，利用对象池技术提高完成端口模型对内存资源的利用效率，采用零字节投递处理重叠I/O的方法降低操作系统资源开销。而对于用户命令消息的传递，为不影响数据传输效率，在单独的线程中以完成例程模型进行管理。

**关键词**：端口转发；完成端口；完成例程；IOCP端口模型；对象池

## 1 内网转发器设计思路

内网转发器与多个客户分别建立多个连接通道，要求有一种高效处理并发连接的模式，多线程模型处理并发的方式是：为每一个连接创建一个线程，让这个线程专门处理该连接的数据I/O。该模型在连接数较多时，会出现以下问题：（1）大量客户连接的建立，关闭，导致线程不断创建，销毁，系统资源开销很大。（2）连接太多导致大量线程同时运行，内核要花费较多时间用于线程间的切换，而每个线程的I/O处理速度则大幅度下降。采用线程池可以解决线程过多的问题。预先创建好一定数量的线程置入线程池，需要时从线程池中取出闲置线程使用，省去创建线程的时间，线程结束运行后，无需销毁，再次放回线程池备用。

简单的线程池模型处理套接字I/O时，在某些场景中仍有线程资源利用不够充分的问题。例如，当某个连接建立后，该连接上请求和应答之间有较长的时间间隔。或者一次请求数据因网络原因分多次传入，只要该连接未断开，会话（session）未结束，线程只能等待会话完成，不能放回到线程池再利用。因此，效率更高的线程模型应该把线程服务的最小单元由一个session分解为单次数据I/O。只要一次I/O结束，线程的任务即完成，可以交回线程池备用。这样，每一个线程的利用率大幅度提高了。微软公司在Winsock2平台上中提供了一个运行于系统内核级别的完成端口模型（Completion Port），实现了上述高效率的多线程I/O管理机制。内网转发器将建立一个基于完成端口模型的数据转发系统。

## 2 完成端口和完成例程

### 2.1 完成端口和重叠I/O

完成端口模型有如下几个特点：

（1）重叠I/O，重叠的含义是不同套接字上数据的接收和发送可以交给内核同时进行，使用WSARecv，WSASend函数完成，以WSARecv为例：

```
int WSARecv(
 _in  SOCKET s,
 _in_out  LPWSABUF lpBuffers,
 _in DWORD dwBufferCount,
 _out LPDWORD lpNumberOfBytesRecvd,
 _in_out LPDWORD lpFlags,
 _in LPWSAOVERLAPPED lpOverlapped,
        _in  LPWSAOVERLAPPED_COMPLETION_ROUTINE
lpCompletionRoutine
);
```

其中第6个参数lpOverlapped是一个称为“重叠结构”的参数，该参数若不为NULL，则表明该函数以“重叠”的方式从套接字s上接收数据。其含义是，WSARecv调用后立即返回，并不阻塞等待数据接收完成，而把接收任务交给系统内核，系统一旦收到数据，则将数据拷贝到由第二个参数lpBuffers指明的一个数据缓冲区，然后以一定方式通知调用者：数据接收已经完成，调用者接到通知后，就可以处理缓冲区里的数据了。可以看出：WSARecv实际上只发出接收数据的指令。调用后立即返回，再次调用，服务于其他套接字，因此可以在短时间内处理大量的并发请求。

（2）I/O完成通知队列和工作线程池。系统处理数据I/O完成以后，把I/O完成包按完成的时间顺序，放入一个通知队列。这个队列由完成端口来管理。完成端口创建若干个工作线程（WorkerThread），所有的工作线程均以调用函数GetQueuedCompletionStatus开始，该函数查询通知队列，取出队列中的完成包，进行数据处理。若队列中为空，则该函数阻塞不返回，工作线程进入挂起（Suspend）状态，直到队列中完成包的到来。多个工作线程挂起等待完成包时，完成端口规定，由最后一个进入等待的线程优先取得完成包。

建立完成端口模型的过程并不复杂，主要流程是：

① 建立完成端口。第一次调用函数：

```
CreateIoCompletionPort(
    HANDLE FileHandle,  //套接字句柄
    HANDLE ExistingCompletionPort, //已经存在的完成端口
    ULONG_PTR CompletionKey, //与套接字相关连的数据
//结构指针
    DWORD NumberOfConcurrentThreads // 一个完成端口
//允许运行的最大工作线程数
);
```

本程序中，建立完成端口的方法是：

```
iocpport = CreateIoCompletionPort(INVALID_HANDLE_VALUE, NULL, 0, thdsum );
```

头三个参数均为 NULL 值，第四个参数根据系统总体设计可以取不同的值，一般等于 CPU 数量

② 建立连接套接字。

③ 把套接字绑定到完成端口，仍然调用函数 CreateIoCompletionPort。

```
CreateIoCompletionPort ((HANDLE)m_sock, iocpport, (DWORD)m_sock, 0);
```

第一个参数设为要绑定的套接字 m_sock，第二个参数为建立好的完成端口 iocpport，第三个参数也同样设为 m_sock。

④ 在工作线程或者其他线程里，调用 WSARecv 或 WSASend 投递 I/O 请求。

⑤ 在多个工作线程里，调用 GetQueuedCompletionStatus 查询完成事件通知队列，取出完成包，进行数据处理。

### 2.2 完成例程模型

完成例程是另一种管理重叠 I/O 的机制，其数据接收和发送也是采用重叠 I/O 函数 WSARecv，WSASend，与完成端口不同的是，在 I/O 请求完成以后，系统将自动调用一个回调函数来处理完成包里的数据。回调函数由用户自己编写，在 WSARecv 函数的第七个参数 lpCompletionRoutine 指明该回调函数的函数指针。完成例程处理多连接的性能比完成端口稍差。内网转发器以完成例程机制来管理用户数据。

## 3 内网转发器系统设计

### 3.1 主线程

内网转发器由主线程、数个完成端口工作者线程和用户管理线程构成。程序流程如图 1 所示。

主线程的工作流程是：（1）创建完成端口，数个工作者线程。（2）创建用户管理线程。（3）开始循环查找外网用户发送的 IP 地址邮件。（4）取出邮件中的 IP 地址，与 外网用户建立连接，建立一个新的用户对象加入用户对象集。（5）以事件通知唤醒用户管理线程去处理新用户。（6）等待用户线程处理完毕通知。（7）继续循环查找用户 IP 地址邮件。主线程代码如下：

```
void  ServerStart(DataModule* pM)
```

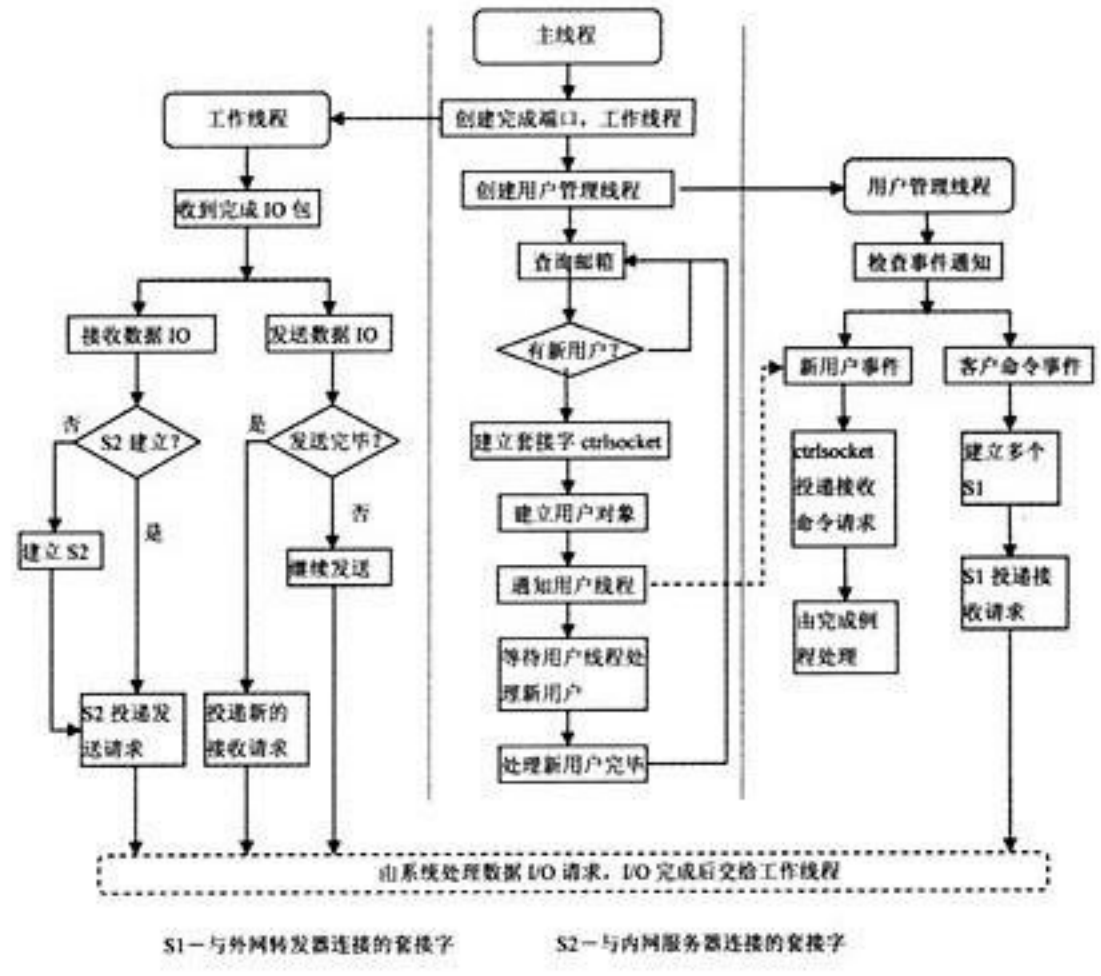



图 1 内网转发器程序流程图

```
{
    int index,ret;
    int poppos=0;
    pM->CreateIOCP(); //建立完成端口和工作线程
    pM->create_event_thread(); //建立用户管理线程
    pM->sendmsg("开始接收邮件 \n");
    pM->runstate = pM->eRUN;
    int servernum = pM->poparray.size();
    if(servernum == 0 ) {
        pM->sendmsg("没有邮件服务器,运行停止! \n");
        return;
    }
    pM->sendmsg("共有%d 个邮件服务器 \n",servernum);
    while(pM->runstate == pM->eRUN )
    {
        pM->popserver= pM->poparray[poppos].pop;
        pM->emluser  = pM->poparray[poppos].user;
        pM->emlpwd   = pM->poparray[poppos].pwd;
        poppos++;
        if(poppos ==  servernum) poppos=0;
        if( !  LookupOuterAddress(pM) ) { //调用查找外网用
//户函数
            WaitForSingleObject  ( pM ->exitevent,  M ->
mailinterval*1000/servernum);
        }
        else {
            for(int i=0;i<pM->emliparray.size();i++)  {
                if( pM->userset.UserExist(pM->emliparray[i])
)
```

```
                    continue;
                pM->ctrlsocket = create_socket() ;
                SetSocketNonBlocked( pM->ctrlsocket );
if(!  connect_delay(pM->ctrlsocket,pM->emliparray[i],2) ) {
                    closesocket(pM->ctrlsocket); //连接失败
                }
                else {
                   pM->sendmsg(" 外网地址连接成功!!! \n") );
                     SetSocketKeepAlive  (  pM ->ctrlsocket,
PULSE); //发心跳包
                          pM ->cur_loguser  =  pM ->userset.
CreateUser( pM->emliparray[i], pM->ctrlsocket );
                    WSASetEvent(pM->acptevent3); //唤醒用
//户管理线程
                    WaitForSingleObject   (pM   ->userevent,
INFINITE); //等待线程处理完毕
                  }
               }
        }
    }
    printf( " 服务停止运行! \n");
}
```

### 3.2 重叠 I/O 数据结构

函数 WSARecv () 调用时，都要传递一个重要的 “重叠结构” (Overlapped) 参数，它给每一个 I/O 操作分配 ID 号作为标志以区分不同的 I/O，在 I/O 操作被系统异步完成后，系统又把这个结构的指针放入 I/O 完成通知队列，重叠数据结构除第一个成员必须为 OVERLAPPED 结构以外，还可以添加其他成员，因此该数据结构可以作为程序和系统之间关于某个 I/O 操作的共享数据区。本系统的重叠结构类型设计如下：

```
typedef struct _PER_IO_DATA
{
          OVERLAPPED  m_Overlapped;   // 重叠结构(每次
//O 操作,该结构都要置零)
     SOCKET   m_sock1;   // 负责接收数据的 Socket
     SOCKET  m_sock2;  // 负责发送数据的 Socket
   char   buf[BUFSIZE];   // 接收数据缓冲区
   int  index; // 对象索引
   WSABUF  wsaBuf; // 结构类型,给 WSARecv 传递数据缓
//冲区参数
     OPERATION   optype;   // 标识操作的类型
     DWORD  bytesend;  //已发送字节数
     DWORD  byterecv;  //已接收字节数
     DataModule*  pM;   //转发系统对象指针
     USER*   pUser;  //关联用户对象指针
} PER_IO_DATA, *PPER_IO_DATA;
```

该数据结构又称为单 IO 数据结构，服务于一次 I/O 操作。

### 3.3 工作线程

数据转发工作由完成端口工作线程进行。工作线程代码如下：

```
UINT WINAPI DataModule::WorkThread( LPVOID lpParam)
{
  THREADPARA* para =   (THREADPARA*)lpParam;
  DataModule *pM = para->pM;
  int  ThdNo = para->nThreadNo;
  HANDLE   iocpport = para->iocpport;//取得完成端口句柄
  DWORD   byteTran = 0;
  PER_IO_DATA    *pIO = NULL;
  SOCKET  hSocket;
  bool  bReturn;
 while(pM->runstate == pM->eRUN)
{
        bReturn  =  GetQueuedCompletionStatus  (  iocpport,
&byteTran,
          (LPD WORD)&hSocket,  (LPOVERLAPPED*)&pIO,
INFINITE);
  if ( 0==hSocket ) break; //收到退出线程命令
  if(bReturn)
  {
    if( byteTran >0 )
    {
      if (pIO->byterecv == 0)       {  // 是接收数据完成包
          pIO->byterecv = byteTran;
          pIO->bytesend = 0;
      }
   else    pIO->bytesend += byteTran;  //发送数据完成包
    //接收字节大于发送字节,说明接收数据未发送或未发送完
//毕,继续发送
   if (pIO->byterecv > pIO->bytesend)  {
   if(pIO->m_sock2 == INVALID_SOCKET) {  //内网服务尚
//未连接
          pIO->m_sock2=create_socket();
          if  (  !  connect_delay  (pIO->m_sock2,pIO->
pUser->loaddr,3) ) {
              printf(" loaddr connect  fail!  \n");
              closesocket( pIO->m_sock2);
                  pIO->pUser->msgbuf[0]='F';
//通知客户端,转发目标连接失败
              send  (pIO->pUser->ctrlsocket,pIO->
pUser->msgbuf,16,0);
          }
          PER_IO_DATA*  pIO2  =  pM->IOPool.GetUnit
(); //建立连接内网的 I/O 结构
          pIO2->m_sock1 = pIO->m_sock2;
          pIO2->m_sock2 = pIO->m_sock1;
          pIO2->pM  = pM;
          pIO2->pUser = pIO->pUser;
          pIO2->optype = ORECV;
           pM ->BindIOCP  (pM ->iocpport2,  pIO2 ->
m_sock1 );
          if( !pM->PostRecv( pIO2, 4096,NULL ) ) //第一
```

```
//次投递接收内网数据请求
                        pM->handle_error(pIO2);
                pIO->pUser->connectsum--;
            }
          //发送已接收的数据
                if ( ! pM->PostSend ( pIO,NULL ) ) pM->
handle_error(pIO);
      }
  else  {   //说明接收数据发送完毕了,可以再次接收数据
            pIO->byterecv = 0;
            if (!  pM->PostRecv ( pIO, BUFSIZE,NULL ) )
pM->handle_error(pIO);
      }
    }
    else {     // BytesTran==0 连接已断开了
  if( pIO->wsaBuf.len == 0){  //处理 0 字节 WSARecv 请求
          if(!  pM->PostRecv( pIO, BUFSIZE,NULL ) )
                    pM->handle_error(pIO);
        }
        else  pM->handle_error(pIO);
    }
  }
  else { // 与 if(bReturn) 对应
    if(pIO == NULL)   continue;
    pM->handle_error(pIO);
  }
 }
   return 0;
}
```

工作线程中，PostSend 函数投递发送数据请求，PostRecv 函数用于投递接收数据请求：

```
bool DataModule::PostRecv ( PER_IO_DATA* pIO ,int buflen
,Rou callfunc )
{
      DWORD dwFlags = 0;
    DWORD dwBytes = 0;
    pIO->bytesend  =0;
    pIO->byterecv  =0;
    pIO->wsaBuf.buf = pIO->buf;
     pIO->wsaBuf.len = buflen;
    pIO->ResetOverlap();
    // 在 m_sock1 上投递 WSARecv 请求
    int r = WSARecv( pIO->m_sock1, &pIO->wsaBuf,1,
&dwBytes, &dwFlags, &pIO->m_Overlapped, callfunc );
    if ((SOCKET_ERROR == r) && (WSA_IO_PENDING ! =
WSAGetLastError())) {
          printf ("WSARecv  Error!   % d,index  % d  \n",
WSAGetLastError(),pIO->index);
        return false;
    }
    return true;
}
```

工作线程循环调用函数 GetQueuedCompletionStatus 查询完成通知队列，函数第一个参数指明要查询的完成端口，当函数返回时，说明有 I/O 操作已经完成，系统把已接收或者已发送的字节数传递给第二个参数 byteTran，套接字句柄传递给第三个参数 hSocket，与该操作关联的单 IO 结构指针传递给第 4 个参数 pIO。

工作线程的数据转发流程固定为在单 IO 结构中的套接字 m_sock1 上接收数据，m_sock2 上发送数据。对于请求数据流，由连接外网客户的 Socket（简称外连接 Socket）接收，由连接内网服务器的 Socket （简称内连接 Socket）发送，共用一个单 IO 结构 pIO，m_sock1，m_sock2 应赋值为：

```
pIO->m_sock1  =  外连接 Socket;
pIO->m_sock2 =   内连接 Socket;
```

而对于应答数据流 ，是由内连接 Socket 接收，再交给外连接 SOCKET 发送，共用一个单 IO 结构 pIO2。因此 pIO2 中 m_sock1，m_sock2 的赋值要变为：

```
pIO2->m_sock1 = 内连接 Socket= pIO->m_sock2;
pIO2->m_sock2 = 外连接 Socket= pIO->m_sock1;
```

为了确保转发的网络数据包不乱序，工作线程采取的转发策略是：一次接收的数据必须全部正确发送完毕后，再进行下一次的数据接收，开始接收数据之前，pIO 中的 byterecv 和 bytesend 全部初始化为 0，数据接收完成后，要将 pIO->byterecv = byteTran，所以可以由 byterecv 是否为 0 来判断 IO 操作是接收还是发送。

单 IO 结构中 m_sock1，m_sock2 的建立时机并不相同，建立外连接要在转发工作开始之前，先由外网转发器发出建立外连接指令，再由内网转发器在处理用户指令的回调函数 UserRoutine 中，循环调用 server_connect_remote (USER* puser) 建立多个外连接 Socket：

```
bool  DataModule::server_connect_remote(USER* puser)
  {
      SOCKET sockets1=create_socket();
      connect_delay(sockets1,puser->readdr,3) ) ;
      PER_IO_DATA* pIO = IOPool.GetUnit();
      pIO->m_sock1 = sockets1;
      pIO->pM = this;
      pIO->pUser = puser;  //与用户关联
      pIO->optype = OSEND;
      BindIOCP(iocpport, pIO->m_sock1 );//套接字绑定到
//完成端口 iocpport
    PostRecv( pIO, 0,NULL )  // 投递 0 字节 recv 请求
      ...
  }
```

函数 connect_delay 在新创建的套接字 sockets1 上建立与外网用户的连接。连接成功后，构建一个单 IO 结构 pIO，pIO->

m_sock1 赋值为套接字 sockets1，把 m_sock1 绑定到完成端口上，最后调用 PostRecv 函数投递一个接收数据请求，让系统等待由 sockets1 传来的请求数据。那么，对内网服务器的连接何时建立？当外网请求数据没有到来时，如果建立与内网服务器的连接，这时由于没有数据交互，连接往往会被服务器关闭而失效。因此，内连接的建立时机只能是在外网的请求数据第一次到来之后发起。工作线程中，当 pIO->m_sock1 收到数据，在向 pIO->m_sock2 发送数据之前，先判断 pIO->m_sock2 是否为 NULL 值，若是，表明与内网服务器的连接尚未建立，则首先在 pIO->m_sock2 上与内网建立连接，再发送请求数据。

与内网服务器建立连接的同时，也要马上准备接收内网服务器传回的应答数据，构建新的单 IO 结构 pIO2，将 pIO2->m_sock1（和 pIO->m_sock2 是同一个套接字，连接内网）绑定到完成端口，对 pIO2->m_sock1 调用 PostRecv，投递接收应答数据的请求。

### 3.4 用户管理线程

为了把转发功能和用户管理分开，使两者的数据传输互不影响，用户管理模块由 UserThread 线程和回调函数 UserRoutine 组成，基于“完成例程”的重叠 I/O 模型构建。代码如下：

```
UINT WINAPI DataModule::UserThread( LPVOID lpParam)
{
  DataModule *pM =  (DataModule *)lpParam;
  WSAEVENT EventArray[1];
  EventArray[0] = (WSAEVENT) pM->acptevent3; //新用户
//到达事件,由主线程通知
  DWORD  idx;
  while(pM->runstate == eRUN) // 主循环
  {
     while(true) //事件通知等待循环
     {
          idx  =WSAWaitForMultipleEvents  (1,EventArray,
FALSE,WSA_INFINITE, TRUE);
        if (idx == WSA_WAIT_FAILED) {
             printf  ("WaitForMultipleEvents  error  % d\n",
WSAGetLastError());
             return FALSE;
        }
        if (idx ! = WAIT_IO_COMPLETION)    break;
     }
     if( pM->runstate == eSTOP )    break;
                  WSAResetEvent  (EventArray  [index  -
WSA_WAIT_EVENT_0]);
     PER_IO_DATA* pIO = pM->IOPool.GetUnit(); //从对象
//池中取出一个闲置对象
     pIO->pUser =  pM->cur_loguser;
     pIO->m_sock1 =  pM->cur_loguser->ctrlsocket;
     pIO->optype  =  ONULL;
     pIO->pM =  pM;
     if(!  pM->PostRecv( pIO, MSGSIZE,pM->UserRoutine
)) //交给完成例程回调函数
          pM->handle_error(pIO);
     SetEvent( pM->userevent ); //通知主线程处理完毕
  }
  return 0;
}
```

回调函数用于控制命令信息处理，代码如下：

```
void CALLBACK DataModule:: UserRoutine(DWORD Error,
                        DWORD byteTran,
                        LPWSAOVERLAPPED Ovlap,
                        DWORD Flags)
{
   PER_IO_DATA   *pIO =  (PER_IO_DATA* ) Ovlap;
   if (Error ! = 0)      {
     pIO->pM->handle_error(pIO);  return;
   }
   if (byteTran == 0)      {
     if(  pIO->wsaBuf.len == 0){ //处理 0 字节 recv 等待
          if  (!  pIO ->pM ->PostRecv  ( pIO, MSGSIZE,
UserRoutine ) )
               pIO->pM->handle_error(pIO);
          return;
     }
      pIO->pM->handle_error(pIO);
      pIO ->pM ->sendmsg  (" 用户  % s  退出 \n",getipstr
(pIO->pUser->readdr) );
      return;
   }
    string name,pwd;
    if( pIO->buf[0]=='#' ) {
        trim(name.assign( pIO->buf+1,9));
        trim(pwd.assign( pIO->buf+10,6));
            //检查用户传来的密码 如果不是有效用户,关闭
//ctrlsocket
        if( !  pIO->pM->userset.IsValidUser( name,pwd) )
{
            shutdown(pIO->m_sock1,SD_BOTH);
        }
        else {      //给 user 对象填上 name 和 password
          strcpy(pIO->pUser->name,name.c_str());
          strcpy(pIO->pUser->pwd,pwd.c_str());
        }
    }
     else if( pIO->buf[0]=='+' ) { //客户端请求建立备
//用连接
        printf(" UserRoutine recv + \n");
         int  c  =  pIO ->pM ->maxconnect  -  pIO ->
pUser->connectsum;
        for(int i=0; i<c;i++) {
```

```
                if (pIO ->pM ->server_connect_remote
(pIO->pUser) ) {
                        pIO->pUser->connectsum++;
                        pIO->pUser->totalcon++;
                }
        }
    }
    else if( byteTran==16) {
            //将接收到的转发目标拷贝至用户对象 pUser->
//loaddr
    memcpy( (char*)&pIO->pUser->loaddr,pIO->buf,16);
            pIO->pM->sendmsg ( "%s 修改转发目标:",
getipstr(pIO->pUser->readdr) );
            pIO->pM->sendmsg( "IP %s  Port %d \n",
                    getipstr(pIO->pUser->loaddr),
          ntohs(pIO->pUser->loaddr.sin_port));
            pIO->pUser->msgbuf[0]='1';  //发送用户超时限
//制大小秒
            memcpy ( pIO ->pUser ->msgbuf +1, (char*)
&pIO->pM->idletimeout,sizeof(int));
    send(pIO->m_sock1,pIO->pUser->msgbuf,16,0);
     }
     if(!  pIO->pM->PostRecv( pIO, MSGSIZE,UserRoutine
) )
            pIO->pM->handle_error(pIO);
}
```

用户管理线程由两个循环构成，内循环不断调用WSAWaitForMultipleEvents检测是否有事件通知，检测的通知事件包括两个，重叠I/O完成通知，该事件通知由系统传递；函数第二个参数指出的用户设定的事件数组。事件数组中只有一个事件pM->acptevent3，该事件用于通知新用户到达，由主线程收到新用户时触发。当该函数返回时，根据返回值判断事件类型；如果返回值为WAIT_IO_COMPLETION，则为重叠I/O完成通知事件。线程运行将中断，由系统转去调用回调函数UserRoutine处理收到的用户命令信息。UserRoutine调用结束，系统返回事件通知循环。如果函数返回值不等于WAIT_IO_COMPLETION，则为acptevent3事件通知，程序跳出内循环到用户处理循环，构建一个PER_IO_DATA对象，将操作类型置为pIO->optype = ONULL，表明该对象只用于接收命令消息，设pIO->m_sock1 = pM->cur_loguser->ctrlsocket，后者为主线程中与外网用户的第一个连接，在pIO->m_sock1上投递接收命令消息请求，注意此处pIO->m_sock1没有绑定到完成端口，不由工作线程处理。PostRecv的第三个参数pM->UserRoutine，指明WSARecv完成后的回调函数。

## 4 完成端口性能优化

### 4.1 对象池技术

重叠I/O模型，每建立一个套接字连接，都要在内存创建相应的I/O结构对象，作为WSASend或WSARecv的参数，当连接关闭时，I/O对象随之被销毁，系统创建释放对象需要开销，而且堆区内存反复的分配释放，会使内存中出现大量的内存碎片，随着时间的流逝，程序运行会越来越慢。可以采用构建对象池的方法解决上述问题，需要时从池中取出一个空闲对象，用完后再放到对象容器中以供下一次再利用。省却了内存分配释放的系统开销；放回对象池的对象在内存中的位置并没有变化，仅仅是内容被重置，不会产生内存碎片。为此设计了一个通用的对象池模版类，代码如下：

```
template <class T> class Pool
{
    CLock  guard;
    deque  <T*>  allunit;
    deque  <T*>  freeque; //闲置对象指针
    public:
    T*   GetUnit ()
    {
        T* p;
        guard.Lock();
        if( freeque.size()>0 ) {
            p= freeque.front();
            freeque.pop_front();
        }
        else{
            p = new T;
            allunit.push_back(p);
        }
        guard.Unlock();
        return p;
}
void  FreeUnit ( T* p)
{
    ((T*)p)->Reset(); //对象被重置
guard.Lock();
    freeque.push_back(p); //加入闲置对象容器里
    guard.Unlock();
}
...
void ResetAllUnit()
    {
        guard.Lock();
        for(int i=0;i< allunit.size();i++)
            ((T*)(allunit[i]))->Reset();
        guard.Unlock();
    }
};
```

该类有一个deque类型的stl容器 freeque保存闲置对象指针；deque容器的优势是，在容器的头尾插入和删除元素的速度比较快。成员函数GetUnit ()调用front ()从freeque首部

取得头部闲置对象指针，取出后，用 pop_front () 函数将该指针弹出，如果没有闲置对象，则在堆区分配一个新对象返回。FreeUnit (T* p) 释放对象，把对象指针 p 从 freeque 尾部加入，两个操作均要用到临界区保护，以确保其他线程互斥访问容器 freeque。另外，模版类实例化的类要实现自己的成员函数 ReSet () 用来重置对象。

程序中的 IO 结构对象池可以这样构建：

```
Pool<PER_IO_DATA>  IOPool;
```

从获取 IOPool 中取得一个闲置的 PER_IO_DATA：

```
pM->IOPool.GetUnit();
```

### 4.2 零字节投递防止内存溢出

投递 WSARecv 请求时，WSARecv 指定的缓冲区会被锁定到系统非分页内存以等待数据，最小为 4k，转发系统要为每个客户预先投递数十个 WSARecv 来等待请求，客户较多时就会有大量的非分页内存被锁定，系统非分页内存的大小是有限的，消耗过多，I/O 操作往往会出现“WSAENOBUFS”的错误。解决的办法是 WSARecv 投递一个 0 字节长度的缓冲区。这样就不会有内存被锁定了，当系统收到数据，会发出一个通知到完成端口，实际并没有接收到数据。可以根据这个通知，再次投递非 0 字节缓冲的 WSARecv 请求，真正开始接收数据。PostRecv 函数第二个参数为缓冲区长度，设为 0，投递 0 字节请求。相应的在工作线程中，函数 GetQueuedCompletionStatus (iocpport,&byteTran…) 第 2 个参数 byteTran 返回 0 字节，但是，当套接字连接断开时，byteTran 也会返回 0，可以通过判断 pIO->wsaBuf.len 是否为 0 来区分这两种情况。

```
…
// bytesTran==0 处理模块
else {
        if( pIO->wsaBuf.len == 0){  //说明是 0 字节
//WSARecv 请求
            if  (!  pM->PostRecv  ( pIO, BUFSIZE,
NULL ) ) //再次投递
                pM->handle_error(pIO);
    }
    else   pM->handle_error(pIO); // 连接已断开了
}
```

## 5 结语

只要是基于 TCP 协议的服务均可经本系统转发。现以外网某主机连接内网的几个服务为例说明：

（1）Http 服务。局域网内某主机 10.10.1.32 有 Web 服务器，在该主机上运行内网转发器；外网主机地址 58.59.3.100，启动外网转发器，监听两个端口 6000,7000。设定转发目标为 10.10.1.32：80，当提示“与内网已经建立！”时，开启浏览器，地址栏输入：“http:// 127.0.0.1:7000/”，即可连入内网 Web 服务器。

（2）Https 服务。Https 也是建立在 TCP 协议上。局域网内某主机 10.10.1.33 运行 RemotelyAnywhere，端口 2000。外网转发器上设定转发目标为 10.10.1.33：2000。浏览器地址栏输入：“https:// 127.0.0.1:7000”，可连入内网 RemotelyAnywhere 服务。

（3）SQLserver 服务器运行于局域网 10.10.1.34,端口 1433。外网转发器设定转发目标为 10.10.1.34:1433，外网数据库客户端修改连接字串中的 Data Source=“127.0.0.1,7000” 可连接内网 SQLserver。

## 参考文献


[1] Jones A，Ohlund J. Windows 网络编程 [M]. 北京：清华大学出版社，2002.

[2] 吴永明，何迪. 基于完成端口的服务器底层通信模块设计 [J]. 信息技术，2007，Vol3.

[3] 程思，程家兴. VPN 中的隧道技术研究 [J]. 计算机技术与发展，2010，20 (2).


（收稿日期：2012-02-25）

（上接第 65 页）

```
    m_WaveOut.Play(bufReceive,SIZE_AUDIO_FRAME);
    m_INIP.SetWindowText(chIPin);
}
```

#### 3.3.2 发送数据代码

```
//发送数据
void CUdp::Sendme(BYTE *data,int len,char *ip)
{
   //发送数据
      SendTo(data,len,port,ip);
}
void CAccDlg::SendAudio(BYTE *pData,int len)
{
    m_Udp.Sendme(pData,len,chIPout);
}
```

## 4 结语

基于 Windows 的低层音频服务、UDP 用户数据报协议开发了一个简单的点对点语音通信程序，介绍了语音通信原理并给出了相关功能的实现代码，对于开发其他类似的应用软件具有参考价值。

（收稿日期：2012-01-13）

# Java class 加密技术研究与实现

任立强

**摘　要**：介绍了利用 Java 虚拟机工具接口进行 class 的加密保护原理，这种保护方法有破解 Java 的平台无关性，在不同的平台下需要现实平台相关的保护代码。

**关键词**：加密；JVMTI 技术；字节码

## 1　引言

Java 语言以其诸多优势，赢得了越来越多的程序员的青睐，但其跨平台特性使得 Java 源代码被编译器翻译成一种中间代码——字节码。class 字节码文件主要存储了在 Java 虚拟机上解释运行的虚指令，未做保护的 Java 字节码文件很容易被反编译成可读性很高的源代码。所以，如何有效地保护 Java 字节码文件不被反编译成源代码是每个 Java 程序员所关心的问题。

Java 虚拟机工具接口（Java Virtual Machine Tool Interface，JVMTI）提供了一种编程接口，允许软件开发人员创建软件代理以监视和控制 Java 编程语言应用程序。下面将介绍到的就是利用 JVMTI 的代理功能，实现 class 文件在加载到内存后，生成 class 对像前对 class 文件内容进行解密。由于这个解密过程不生成临时文件，解密后的数据直接在内存在生成 class 对像，代理的编写又必须使用本地代码，这样，代码的安全性又高了一层。如果给代理 DLL 进行加壳加密以及加虚拟机处里，那安全就更固靠了。

## 2　JVMTI 技术加密 Class 文件原理

Jvmti 是 Java 虚拟机工具接口，它提供了一种编程接口，允许软件开发人员创建软件代理以监视和控制 Java 应用程序。Jvmti 代理程序可从 JVM 接受自己兴趣的事件，进而对事件做出相应的处理。这里正是利用 JVM 事件通知来实现 class 文件在载入虚拟机之前进行解密，这种解密计算完全在内存中实现，解密后的 class 数据也只是在内存中生成 class 对像，不会产生任何临时文件，这对于想窥探你的应用程序秘密的人来说将是一道屏障。下面具体介绍如何实现。

### 2.1　Agent_OnLoad 代理函数

代理程序必须包含一个名为 Agent_OnLoad 的函数，JVM 在加载库时要调用这一函数，代理程序从这里开始：

```
JNIEXPORT jint JNICALL Agent_OnLoad (JavaVM *jvm, char *options, void *reserved)
{
 ……
 return JNI_OK;
}
```

在该函数中，注册所感兴趣的事件，这样，当注册的事件发生时，JVM 会调用事件对应的回调函数。JVMTI 中有很多事件，关键的是 JVMTI_EVENT_CLASS_FILE_LOAD_HOOK，该事件意思是让 JVM 在载入每个 class 文件时触发回调函数。

### 2.2　注册事件

```
JNIEXPORT jint JNICALL Agent_OnLoad (JavaVM *jvm, char *options, void *reserved)
{
gb_jvmti ->SetEventNotificationMode (JVMTI_ENABLE, JVMTI_EVENT_CLASS_FILE_LOAD_HOOK, NULL);
 return JNI_OK;
}
```

### 2.3　定义事件的回调函数

```
JNIEXPORT jint JNICALL Agent_OnLoad (JavaVM *jvm, char *options, void *reserved)
{
jvmtiEventCallbacks callbacks;
callbacks.ClassFileLoadHook = &ClassFileLoadHook;
gb_jvmti->SetEventCallbacks(&callbacks, sizeof(callbacks));
 return JNI_OK;
}
```

这样，就定义了 JVMTI_EVENT_CLASS_FILE_LOAD_HOOK 事件的回调函数 ClassFileLoadHook。

### 2.4　ClassFileLoadHook 函数

ClassFileLoadHook 函数的参数原型及参数介绍：

```
void JNICALL
ClassFileLoadHook(jvmtiEnv *jvmti_env,JNIEnv* jni_env,
jclass class_being_redefined,
                  jobject loader,const char* name,jobject protection_domain,
                  jint class_data_len,const unsigned char* class_data,
```

```
jint* new_class_data_len,unsigned char** new_class_data)
```

可以看到，ClassFileLoadHook 有多个参数。这里只介绍和解密相关的 4 个参数，其他的请参考 jvmti 手册。

jint class_data_len：JVM 从 class 文件读入的原始 class 文件的长度。

const unsigned char* class_data：JVM 从 Class 文件读入的原始 class 文件的内容。

jint* new_class_data_len：解密处理后 class 文件内容的长度。

unsigned char** new_class_data：解密处理后的新内容放这里。

JVM 在每个类载入时，都会调用 ClassFileLoadHook 函数，此时，class_data 和 class_data_len 分别保存了从 class 文件读取的内容和长度，这个文件内容是否被加密是不确定的，类在这个时候也没有在 JVM 中生成 class 对像。当函数执行完后，JVM 会判断 new_class_data_len 和 new_class_data 这两个参数是否被设置了值，如果没有设置，那么 JVM 就用 class_data 和 class_data_len 所保存的值生成 class 对像；如果设置了值，JVM 就用 new_class_data_len 和 new_class_data 生成 class 对像，字节码的有效性验证也是在 classFileLoadHook 这个函数执行完 class 对像生成前进行。

## 3 加密和解密的实现

这样，只需要在 ClassFileLoadHook 函数中完成 class 的解密功能，把加密的 class_data 解密后放入 new_class_data，并计算解密后 class 内容的长度放入 new_class_data_len，生成 Class 对像的工作在你完成解密后将由 JVM 自动完成，无须手工操作：

```
void JNICALL
ClassFileLoadHook(jvmtiEnv *jvmti_env,JNIEnv* jni_env,
jclass class_being_redefined,
        jobject loader,const char* name,jobject
protection_domain,
        jint class_data_len,const unsigned char*
class_data,
        jint* new_class_data_len,unsigned char**
new_class_data)
{
bool bOK = true;
long flen = class_data_len;
if (flen < 0x1c) {//文件长度是否合格
    bOK = false;
}
int foff = 0;
int magic[] ={0xAA,0xBB,0xCC,0xDD};
for (; foff < 4; foff++){//是否被加密过
    if (class_data[foff] ! = magic[foff]) {
        bOK = false;
        break;
    }
}
if (bOK){//文件长度正常且文件被加密过,进入解密过程(在
//这里可以设计自己的解密算法,解密细节不做详细介绍)
    int hlen = 0x18;
    char header[24];
    int headerEnd = hlen + foff;
    int i = 0;
    for (; foff < headerEnd; foff++) {
        header[i] = class_data[foff];
        i++;
    }
    if (header ! = NULL){
        Exchange(header,24);
        //int key[] = {0x87,0x76,0x65,0x54};
char header_key[4] = {(char)0x87,0x76,0x65,0x54};
        char clsKey[24];
        Xor(header_key, 4 , header , 24 ,clsKey);
        char* cfbuf = (char *)malloc(flen-foff);
        if(! cfbuf)
        {
    printf("Memory Allocated Failure! ",cfbuf);
            return;
        }
        i = 0;
        for (; foff < flen; foff++) {
            cfbuf[i] = class_data[foff];
            i++;
        }
        if (cfbuf ! = NULL) {
    char * dcbuf = (char *)malloc(flen-28);
            if(! dcbuf)
            {
        printf("Memory Allocated Failure! ",dcbuf);
                return;
            }
        Xor(clsKey, 24, cfbuf, flen-28 ,dcbuf);
            Exchange(dcbuf, flen-28);
            char cls_Key [16] = { 0x48, 0x59,
(char) 0xAB, 0x3E, 0x76,
        0x74, 0x04, 0x05, 0x79, 0x32, 0x1F, (char) 0x93,
  (char) 0xBC, (char) 0xAC, (char) 0xBA, (char) 0xBE};
        char * new_class_data_temp = (char *)malloc
(flen-28);;
    Xor(cls_Key, 16, dcbuf, flen-28 ,new_class_data_temp);
                *new_class_data_len = (flen-28);
*new_class_data = (unsigned char *)new_class_data_temp;
                /*
```

(下转第 93 页)

# 多远程桌面自动登录助手软件设计与实现

徐东玲

**摘　要**：通过对网络运维中需要频繁登录多个远程主机进行管理的需求分析，利用 C# 开发了多个远程主机从一个界面完成远程登录及浏览的多个远程桌面自动登录助手软件，简化了登录过程，方便了运维操作。

**关键词**：远程桌面；C# 语言；用户名；密码

在与互联网没有交连的相对安全的内网中，网络维护者及部分使用者经常要频繁远程登录多个主机进行管理，但是因为 IP 地址、用户名及登录密码各不相同，造成登录维护的不方便。下文在 Visual Studio 2010 中利用 C# 语言，实现了多个远程操作在一个界面中方便实施的功能，即多远程桌面自动登录助手软件。

## 1　软件功能描述

在专网中维护多个主机，不同主机可能处于不同位置，同时还可能需要定时浏览监控画面、观察 UPS 运行情况或者远程进行电源管理，众多不同的登录 IP 地址、登录密码容易造成混淆，给维护监控带来一定的困难。该软件把需要管理的主机按位置功能进行分组，把需要完成的主要功能操作合理组织、部署在该位置主机的管理界面上，把对应的登录 IP 地址预先初始化在程序里，调用预先保存的登录信息，就可以通过该软件，一键完成对应的登录操作。

## 2　软件设计

### 2.1　软件界面

软件运行界面如图 1 所示。

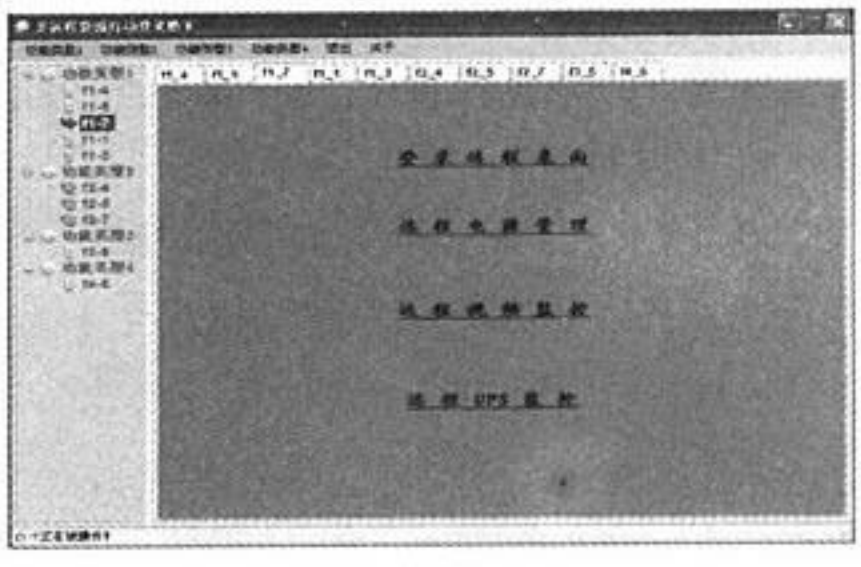


图 1　软件界面

软件把要登录的主机位根据功能进行分组。这里假设共有 4 个功能分组：功能类型 1、功能类型 2、功能类型 3 和功能类型 4。以功能类型 1 为例，它包含 5 个需要维护操作的主机点位 f1_4、f1_5、f1_7、f1_1 和 f1_3。当要维护操作 f1_7 主机时，则点开节点“功能类型 1”分组选择 f1_7，出现 f1_7 主机操作界面；以 f1_7 主机操作界面为例，它需要组织部署“登录远程桌面”、“远程电源管理”、“远程视频监控”和“远程 UPS 监控”4 个功能操作，分别运用功能链接键实现；维护时根据需要点击相应的功能链接键完成对应的操作。选择其他主机功能操作与 f1_7 类似，在此不一一表述。

### 2.2　界面部署

为了方便操作，界面由菜单栏、操作部分和状态栏组成。菜单栏 menuStrip1 可以用来打开各个“功能类型”分组直接进行操作，还包括“退出”程序和“关于”程序简要说明。状态栏 statusStrip1 主要用于简单显示当前操作的结果。操作部分由 treeView1 控件和 tabControl1 控件组成。treeView1 控件用于树型指示各功能类型分组，对正在操作的主机位则以红色箭头进行提示。tabControl1 控件以标签形式把所有需要操作的主机位进行排列，在每个主机位标签里，把需要进行的操作进行了组织，达到了方便操作的目的。具体参看图 1。

### 2.3　窗口初始设置

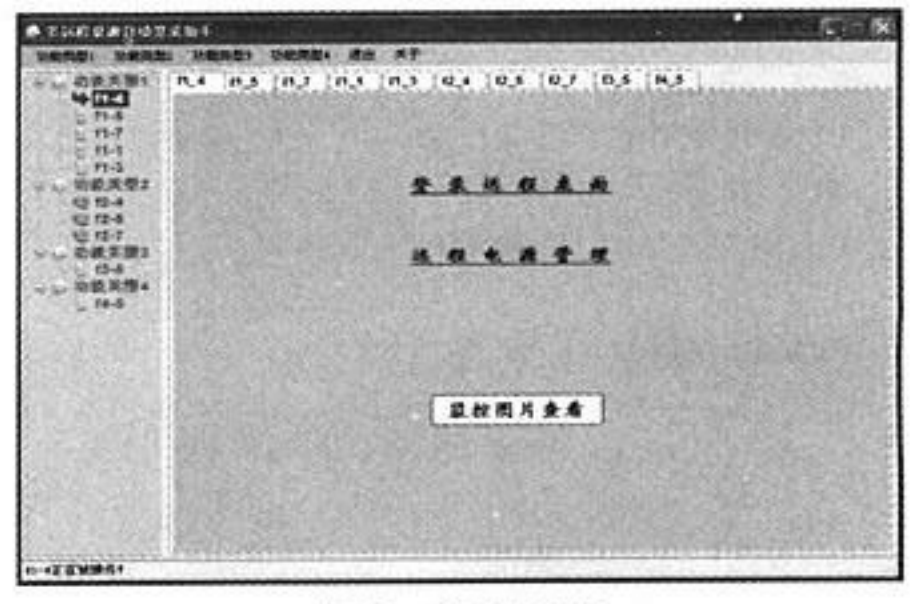


图 2　初始界面

窗体加载时，调用窗体的 Load 函数，首先读取初始化设置文件，更新窗口默认设置，然后设定 treeView1 展开，

treeView1 指示第 0-0 节点为当前操作主机、tabControl1 控件的 TabPage0 为操作焦点，如图 2 所示。

实现代码如下：

```
private void Form1_Load(object sender, EventArgs e)
   {
     iniFileOP(); //初始化方法 1,主要检查初始化 ini 文件是否
//存在
    iniUpdateFrm(); //初始化方法 2,根据 ini 文件更新窗口设置
    treeView1.ExpandAll();
      this.treeView1.SelectedNode  =treeView1.Nodes  [0].
Nodes [0];
    this.tabControl1.SelectedTab = f1_4;//分别设置操作焦点
    string str = treeView1.SelectedNode.Text;
     toolStripStatusLabel1.Text = str+"正在被操作！";//设置状
//态栏
   }
```

### 2.4 需要特别导入的命名空间

```
    using System.IO;
using System.Text;
using System.Diagnostics;
using System.Net.NetworkInformation;
```

### 2.5 选取主机位的表示与关联

当从 treeView1 控件打开某一主机位时，tabControl1 控件同时打开对应的 TabPage 页；同样，当从 tabControl1 控件打开某一主机位时，treeView1 控件也同时指向对应的主机位，以红色箭头指示。实现这种关联，主要通过 treeView1 的 AfterSelect（）方法和 TabControl1 的 Selecting（）方法实现。代码如下：

```
     private void treeView1_AfterSelect (object sender,
TreeViewEventArgs e)
{try
     {
      string str = treeView1.SelectedNode.Text;
      switch (str)
       {
        case "功能类型 1":
         this.tabControl1.SelectedTab = f1_4; break;
        case "f1_4":
         this.tabControl1.SelectedTab = f1_4; break;
         ......
        default: break;
       }
        toolStripStatusLabel1.Text = str + "正在被操作！";
     }
        catch (Exception ee)
         { MessageBox.Show(ee.Message); }
};
        Private void TabControl1_Selecting (Object sender,
TabControlCancelEventArgs e)
{
switch (e.TabPageIndex)
{
  case 0:
    treeView1.SelectedNode = treeView1.Nodes[0].Nodes[0];
    break;
  case 1:
    treeView1.SelectedNode = treeView1.Nodes[0].Nodes[1];
    break;
  case 2:
    treeView1.SelectedNode = treeView1.Nodes[0].Nodes[2];
    break;
     .....
    default: break;
 }
 string str = treeView1.SelectedNode.Text;
 toolStripStatusLabel1.Text = str + "正在被操作！";
}
```

### 2.6 主要功能链接键

主机位操作，“远程 UPS 监控”和“远程电源管理”等主要是登录具体 IP Web 网址进行，操作以点击 LinkLabel 按钮调用 Process 命令实现。代码如下：

```
     private vo id linkLPMf1_7_LinkClicked (object sender,
LinkLabelLinkClickedEventArgs e)
        {
          try
          {
            Process.Start("http://192.168.1.54/index.htm");
          }
          catch (Exception ex)
          {
             MessageBox.Show(ex.Message, "ex");
          }
        }
```

“登录远程桌面”是本软件的主要功能，也是点击 LinkLabel 按钮完成，代码实现根据使用要求有些差异。首先需要调用 ping 命令，探测一下主机是否开机。如果开机，则再用 Process 实现“mstsc.exe”远程登录；如果没有开机，则弹出提示窗口。同时 ping 命令结果在状态栏显示。代码如下：

```
     private void linkLRDf1_7_LinkClicked (object sender,
LinkLabelLinkClickedEventArgs e)
   {
     try
     {
       Ping PingInfo = new Ping();
       PingOptions PingOpt = new PingOptions();
       PingOpt.DontFragment = true;
       string myInfo = "hyworkhyworkabcdefghijklmnopqrstop";
       byte[] bufferInfo = Encoding.ASCII.GetBytes(myInfo);
       int TimeOut = 120;
```

```
    PingReply reply = PingInfo.Send("192.168.1.7", TimeOut,
bufferInfo, PingOpt);
    if (reply.Status == IPStatus.Success)
    {
    toolStripStatusLabel1.Text = "成功转接 192.168.1.7";
      Process.Start ("mstsc.exe", "\"C:\\Documents and
Settings\\Administrator\\My Documents\\f1_7.rdp\"");
    }
    else
    {
  toolStripStatusLabel1.Text = "连接 192.168.1.7 不成功";
     MessageBox.Show("无法 Ping 通");
    }
   }
   catch (Exception ey)
   {
     MessageBox.Show(ey.Message);
   }
  }
```

## 2.7 远程自动登录

在 2.6 节中由 Process 调用“mstsc.exe” 实现远程自动登录时，需要区别两种情况进行处理。

一是管理主机为 Windows Server 2003 系统，远程主机是 Windows XP 系统。可以把登录远程主机的 IP、用户名和密码在 2003 系统下预存在某个文件夹下，如 My Documents 里，存储的形式为.rdp 文件，如 f1_7.rdp 文件里存的为远程登录 f1_7 主机用到的 IP、文件名和密码。具体 Process 命令执行代码为：

```
Process.Start ("mstsc.exe", "\"C:\\Documents and
Settings\\Administrator\\My Documents\\f1_7.rdp\"");
```

二是管理主机为 Windows Server 2003 系统，远程主机也是 Windows Server 2003 系统。Process 命令需要加上/console 才能执行，代码如下：

```
Process.Start("mstsc.exe", "/console \"C:\\Documents and
Settings\\Administrator\\My Documents\\ f1_5.rdp\"");
```

三是管理主机为 Windows XP 系统。不能利用系统预先存储远程登录主机的 IP、用户名和密码，需要先打开远程登录界面，再手动填写用户名和密码。具体的登录代码为：

```
Process.Start ("mstsc.exe", "/v 192.168.1.24 /admin /w:
800 /h:600");
```

## 2.8 程序唯一执行

为保证程序执行时的唯一性，在程序执行时，对执行线程进行分析。如果有此程序的线程，则弹出对话框“本程序一次只能运行一个实例”并不再运行；如果没有此程序的线程，则执行操作。具体实现是在程序的入口处加入分析代码，实现代码举例如下：

```
        string MName = Process.GetCurrentProcess
().MainModule.ModuleName;
        string PName = Path.GetFileNameWithoutExtension
(MName);
      Process[] myProcess = Process.GetProcessesByName
(PName);
if (myProcess.Length > 1)
{
MessageBox.Show("本程序一次只能运行一个实例！", "提示",
MessageBoxButtons.OK, MessageBoxIcon.Information);
}
else
{
Application.EnableVisualStyles();  Application.SetCompatible
TextRenderingDefault(false);
Application.Run(new Form1());
}
```

## 3 结语

在部分特殊用户的网络远程维护中，通过多个远程桌面自动登录助手软件的实际应用，达到了方便远程操作、提高维护效率的目的。在特定的网络管理工作中开发该软件有一定的实际应用价值。

## 参考文献

[1] 梁冰，吕双，王小科. C# 程序开发范例宝典. 2 版. 北京：人民邮电出版社，2009，10.

（收稿日期：2012-03-15）

（上接第 68 页）

```
      {
        if(prevbuflen-(*lastpos)==1) //到达最后一个字节
          {
     postbuf [(*postbuflen)]=1; postbuf [(*postbuflen)+1]=
prevbuf[prevbuflen-1];
   postbuf[(*postbuflen)+2]=0;postbuf[(*postbuflen)+3]=0;
     *lastpos=prevbuflen;*postbuflen=4+(*postbuflen);
          }
      }
```

```
    }
  }
```

## 6 结语

程序开发完成，实现了 24 位色转换 8 位色 BMP 的功能。生成的 8 位色 BMP 和 24 位色 BMP 相比，视觉效果不错，转换速度也比较快。读者可以通过阅读源程序，得到更多的相关信息。

（收稿日期：2012-01-16）

**Q 怎样巧用 MediaPlayer 组件获取 MP3 文件信息**

A MP3 文件是现今流行的音乐文件格式，被广泛应用在计算机、音乐播放器等电子设备中，MP3 是一种包含媒体信息的文件格式，换句话说，文件中包含了例如歌手名 (Artist)、专辑名 (Album) 等信息，这些信息都存放在文件中的一个叫 Tag 的区域中，读者如果有兴趣可以参考 http://baike.baidu.com/view/66078.htm 获取有关 MP3 Tag 的详细介绍，通过一些简单的编程，可以从 Tag 中获取 MP3 文件的信息，然而并非所有的信息都可从 Tag 中直接获取，一个典型的例子是 MP3 音乐的持续时间这一重要信息就无法直接从 Tag 中读取。下面介绍一种巧妙利用 MediaPlayer 组件来获取 MP3 信息的方法。

大家都知道，Windows Media Player 是 Microsoft Windows 操作系统自带的媒体播放软件，由于它近似于系统组件，因此可以将其视为操作系统的一部分，Media Player 组件是 Windows Media Player 可以在 Windows 上运行的 DLL 文件，在 Windows 操作系统中，这一 COM 组件存在于 WMP.DLL 文件中，利用该组件，可以在自己的应用程序中使用 Media Player 的各项功能，当然也就包括获取 MP3 文件的信息这一基础功能了。下面说明一下使用 C# 语言实现此功能的具体的编程方法。

首先要在自己的 Project 中添加 COM 组件 WMPLib，如上文所述，这一组件存放在 WMP.DLL 中。添加完毕组件后，VS 的 IDE 的智能感知功能将对 WMPLib 可用，可以使用如下的代码来获取 MP3 文件信息：

```
WMPLib.WindowsMediaPlayerClass player = new WMPLib.WindowsMediaPlayerClass();
String mp3filename="d:\\mp3\\test.mp3";
WMPLib.IWMPMedia media = player.newMedia(mp3filename);
string author=media.getItemInfo("Author").Trim();
string duration=media.durationString;
```

下面对代码进行一些解释，代码的核心是通过一个 IWMPMedia 的接口来获取信息，不过要获得这个接口，必须先创建一个 Media Player 播放器，这就是 WindowsMediaPlayer Class 的作用，通过此类型创建的变量 player 来调用 newMedia 函数即可返回一个 IWMPMedia 接口类型变量 media，可以假想自己在程序中创建了一个虚拟播放器，而 newMedia 函数就相当于让播放器选择了一个 MP3 文件来播放，随后可以使用 getItemInfo 函数来获取有关 MP3 的 Artist,Album,Title 等基础信息，具体获取哪种信息，可以将相关字符串传入到该函数的参数中，例如示例代码中传入了 “Author”，这样将获取该 MP3 的作者信息，这里顺便说明，Author 事实上是 Artist 的别名，具体可以传入哪些字符串请读者参考 MSDN，要获取 MP3 持续时间，则可以使用 durationString 属性，这一属性将直接返回 MP3 时长的字符串。

最后总结一下使用此方案的利弊，使用此方法获取 MP3 信息，好处是编程简单，不仅获取 Author,Album 之类的信息可以免去研究 Tag 结构和直接在自己的代码中读取文件，更重要的是获取诸如 MP3 时长之类的信息可以免去复杂的运算，编程非常简洁；劣势也是显而易见的，从上文代码中看到，要获取 MP3 信息，不得不建立了一个播放器（虽然其不可见，但事实上存在），这无疑是一个很大的开销。好在现在计算机硬件配置已经不是瓶颈，如果不是特别在意这方面的话，各位读者可以参考本文的简洁方案来实现读取 MP3 信息的功能。

（作者：李斌）

**Q 如何实现 Java 调用 C 或 C++函数**

A 在现今的软件开发领域中，Java 以其跨平台的优势得到大量的应用，其代码可以一次编译多处执行。但这种特性给 Java 带来了一定的局限性，幸好 Java 提供了完备的 C/C++语言接口，这样可以利用 C 语言的强大功能实现 Java 难以实现的功能，在一定程序上消除 Java 的局限性和低效率。

（1）创建 DLL 文件

使用某一种 C/C++开发工具创建 DLL 文件，实现某一功能，供 Java 调用，例如本文在此使用 Visual studio 2005 创建一个名为 testdll 的动态库文件。

（2）使用 JNI

JNI 是 Java Na***tive Interface 的缩写，中文为 Java 本地调用。它允许 Java 代码和其他语言写的代码进行交互。

1）Java 类：在 Java 程序中，首先需要在类中声明所调用的库名称，如下：

```
Static
{
System.loadLibrary("testdll"); //加载动态库,testdll 为 DLL //文件名称
}
```

还需要对将要调用的方法做本地声明，关键字为 native。并且只需要声明，而不需要具体实现。如下：

```
public native static void set(int i);
public native static int get();
```

然后编译该 Java 程序文件，生成 CLASS，再用 JAVAH 命令，JNI 就会生成 C/C++的头文件。

例如程序 testdll.java：

```
public class testdll
{
static
{
System.loadLibrary("testdll");
}
```

## TROUBLESHOOTING OF PROGRAM

```
public native static int get();
public native static void set(int i);
public static void main(String[] args)
{
testdll test = new testdll (); test.set (10); System.out.
println(test.get());
}
}
```

用 javac testdll.java 编译它，会生成 testdll.class。

再用 javah testdll，则会在当前目录下生成 testdll.h 文件，这个文件需要被 C/C++程序调用来生成所需的库文件。

2）创建 C/C++项目需要增加的头文件有 jni.h、jni_md.h 这两个文件是 JNI 中必须的；还有就是增加 testdll.h。

对于已生成的.h 头文件，C/C++所需要做的，就是把它的各个方法具体的实现。然后编译连接成库文件即可。再把库文件拷贝到 Java 程序的路径下面,就可以用 Java 调用 C/C++所实现的功能了。

接上例子。先看一下 testdll.h 文件的内容：

```
#include
#ifndef _Included_testdll
#define _Included_testdll #ifdef __cplusplus extern "C"
{
 #endif JNIEXPORT jint JNICALL Java_testdll_get
(JNIEnv *, jclass);
JNIEXPORT void JNICALL Java_testdll_set (JNIEnv *,
jclass, jint);
#ifdef __cplusplus
}
#endif
#endif
```

在具体实现的时候，只关心两个函数原型 JNIEXPORT jint JNICALL Java_testdll_get (JNIEnv *, jclass) 和 JNIEXPORT void JNICALL Java_testdll_set (JNIEnv *, jclass, jint)；这里 JNIEXPORT 和 JNICALL 都是 JNI 的关键字，表示此函数是要被 JNI 调用的。而 jint 是以 JNI 为中介使 Java 的 int 类型与本地的 int 沟通的一种类型，可以视而不见，就当做 int 使用。函数的名称是 JAVA_ 再加上 java 程序的 package 路径再加函数名组成的。参数中，也只需要关心在 Java 程序中存在的参数，至于 JNIEnv* 和 jclass 一般没有必要去碰它。

下面用 testdll.cpp 文件具体实现这两个函数：

```
#include "testdll.h"
int i = 0;
JNIEXPORT jint JNICALL Java_testdll_get (JNIEnv *,
jclass)
{ return i; }
JNIEXPORT void JNICALL Java_testdll_set (JNIEnv *,
jclass, jint j)
{ i = j+5; }
```

编译连接成库文件，这里就是 testdll.dll。把 testdll.dll 拷贝到 testdll.class 的目录下，java testdll 运行它，就可以观察到结果了。

使用 JNI 可以在 Java 中调用其他语言编写的代码，在一定程度上消除 Java 的局限性和低效率。

（作者：闫爱涛）

（上接第 88 页）

```
                FILE *fpp;
        if((fpp=fopen("c:\\aa.class","wb"))! =NULL)
                {
                    long b=cls_len;
            fwrite(new_class_data_temp,1,b,fpp);
                    fclose(fpp);
                }
                */
                if(dcbuf)
                    free(dcbuf);
            }
            if(cfbuf)
                free(cfbuf);
        }
    }
}
```

以上功能代码最终需要编译成本地代码，以动态库形式提供给 Java 虚拟机，动态库的存在就破坏了 Java 的跨平台特性，这也是此方案的局限性。如果应用需要跨平台运行，那么需要为每个平台编写此动态库。

当然，也可以进一步保护动态库以不被反汇编或被动态跟踪。如，在 Windows 下的 DLL 可以加壳，目前的流行加密壳都有虚拟机和反调试器的功能。通过这样保护措施，足可以把绝大多数解密者挡在门外。

## 4 结语

不管那种字节码的保护方法，都会增加应用的系统开销，延长系统启动和运行时间，所以，有选择的保护认为重要代码是有必要的。如果千篇一律地保护应用系统的所有类文件，由此带来的系统开销会让人痛头，甚至于放弃保护措施，使代码暴露在容易受攻击的形势下。

## 参考文献

[1] 使用 Java 虚拟机工具接口（JVMTI）创建调试和分析代理.

[2] JVMTM Tool Interface Version 1.1.

（收稿日期：2012-01-08）

# 电脑系统维护经验与技巧

**如何让机密文件永远消失**

使用 DiskGenlus 可以轻松将文件彻底消失。首先进入 DiskGenlus 主界面，在删除对象上单击鼠标右键，选择快捷菜单中的“彻底删除”选项。然后选择填充方式，比如用 0 填充、用随机数填充等。对存放于 NTFS 分区的数据，要勾选“删除 SMFT 中的文件记录”选项。而对于 FAT/FAT32 分区之中的数据，则要选择“删除目录项”，最后按下“彻底删除”按钮就可以了。由于彻底删除功能的的执行速度相对于普通删除方式来说比较慢，所以在删除容量较大的对象时需要耐心等待。

为了验证删除效果，特别选用了几种数据恢复工具对所删除数据进行恢复，结果均告失败，验证了彻底删除数据的效果是非常理想的。

**怎样使用 DiskGenlus 备份数据**

点击主菜单上“工具”选项，在下拉菜单中选择“备份妥区到镜像文件”。选择备份分区及保存路径，然后按下“开始”按钮，就能够对文件进行备份操作了。

当以后打算还原数据时，不妨选择菜单列表上“从镜像文件还原分区”功能选项，即可轻松搞定。

除了备份分区功能外，在 DiskGenlus 中还有克隆硬盘及分区功能，因实现方法较简单，故在此不再多述。

**如何选购传输速度 6Gbps 硬盘**

6Gbps 硬盘开始在市场上迅速普及。相比 3Gbps 硬盘来说，6Gbps 接口的硬盘最大的改变就是理论传输速度从 3Gbps 提升到 6Gbps。对于追求高速度的电脑用户来说，6Gbps 硬盘绝对是最佳选择。选购经验如下：

(1) 碟片数量

目前来看，硬盘厂商的单碟容量技术普遍达到了 500GB 水平。因此，如果是 1TB 硬盘，那么需采用双碟设计。如果是 2TB 硬盘，最多也只需要 4 碟设计。考虑到现在的硬盘技术已经相当成熟，碟片数量越少，硬盘发生故障的几率也就越低。所以说，选购硬盘的时候，应该尽量选择碟片数量更少的硬盘。除此之外，单碟容量更大的硬盘，其数据读取速度也要更快一些。因此，单碟容量为 500GB 的硬盘，应该是选购 6Gbps 硬盘的最低门槛。

(2) 缓存容量

除了单碟容量之外，缓存容量也是不容忽视的硬盘参数。所谓缓存，其实就是硬盘与外部总线交换数据的场所。硬盘在读取过程中把磁信号转换成电信号，并通过缓存的一次次填充清空、再填充再清空，一步步地近按照总线周期发送出去。所以缓存的作用是不容小视的，大容量高速缓存对硬盘性能会有明显提升。不过需要注意的是，就在 6Gbps 硬盘全面普及的时候，西数旗下的部分产品却在缓存这个参数上出现了“缩水”。以西数绿盘 WD10EARS（3Gbps/64MB）和西数绿盘 WD10EADX（3Gbps/32MB）这两款硬盘为例，在选购时就非常让消费者纠结。但是从评测数据来看，后者虽然缓存容量“缩水”为 32MB，但是凭借着 6Gbps 接口的优势，其性能仍然略微领先。因此消费者在选购时，还是应该首先查看硬盘接口是否为 6Gbps，之后再考虑缓存容量。

(3) 硬盘转速

由于低碳节能的概念越来越普及，所以市场上的低功耗硬盘也越来越多。其中希捷旗下的低功耗硬盘以 LP 最为常见，硬盘转速为 5900rpm。日立旗下的低功耗硬盘则以酷冰系列为主，硬盘转速为 5400rpm。至于西数旗下的西数旗下的低功耗硬盘则以“绿盘”为主，可以实现 5400～7200 rpm 的动态调节。因此，对于普通家庭用户来说，西数旗下的“绿盘”最为适合。而希捷和日立旗下的低功耗硬盘，则仅仅适合于下载和节能用户。

(4) 硬盘型号

现现如今，硬盘的种类越来越多，单就西数硬盘来说，就分为绿、蓝、黑 3 大系列。而在 3 大系列的下面，又分为多个小系列，所以普通消费者几乎很难辨别清楚。因此消费者在选购时，还是应该以性价比为原则，比如说西数硬盘，就应该首选价格最为低廉的“绿盘”系列。

除此之外，普通消费者既然弄不清楚硬盘的所属系列，就干脆记住要买硬盘的型号吧。并且有些时候，硬盘型号也可以很直观地反映出硬盘的各项性能参数。以希捷 ST31000524AS 这款硬盘为例，数字 3 就代表了 3.5 英寸硬盘，数字 1000 代表了 1TB 容量，数字 5 代表了缓存为 32MB（数字 4 和 6 分别代表 16MB 和 64MB 缓存），数字 2 则代表了双碟设计。

**硬盘的内部速度与外部速度有何不同**

硬盘的速度分为内部速度和外部速度。内部速度是指硬盘读写速度，目前主流硬盘读写速度在 60MB/s 至 120MB/s 之间。而硬盘的外部速度是指硬盘与南桥之间的数据传输速度，其实是硬盘缓存与南桥芯片的传输速度，这个速度一般都能达到 150MB/s 至 250MB/s。

**硬盘采用 SATA 接口有何明显优势**

采用 SATA 接口的硬盘的性能比采用 PATA 的同系列硬

## MAIL TO THE DOCTOR

盘高，不仅传输速度快，而且也降低了硬盘传输的错误率。

**？ 为什么移动硬盘在使用中出现无法识别的问题**

！ 如果移动硬盘在使用中系统能够发现新硬件，但不能读盘，识别为“Unknown Device”说明移动硬盘并没有被正确识别，换电脑使用可排除硬盘本身的问题。这种现象可能和移动硬盘的接口及操作系统有关系。建议用户连接时除了连接好外置供电接口外，同时尽可能使用原装线缆，并且不要使用USB延长线。

**？ 移动硬盘的USB连接线接入有否顺序**

！ 移动硬盘在启动时，需要比正常工作时更大的电流，如果这关通过了，进入移动硬盘就不成问题了。在这方面，很多移动硬盘线都用了Y型USB数据线（就是在USB数据接口的基础上，还增添了一个专用供电的USB接口）。

很多用户在这种数据线的使用上并没有特别注意，通常都是随意插拔，很多时候不少用户将移动硬盘的数据线一直连接在移动硬盘上，这样一来，当用户将连接线的第一个USB接头接入电脑时，移动硬盘就进入了启动状态，而此时用户还没有来得及将第二个USB接头插入电脑，就只有一个USB接口在为硬盘启动供电，这将直接导致移动硬盘启动失败。所以当用户使用Y型USB数据线时，可先将数据线的两个USB接头连接到电脑上，再接入移动硬盘。

**？ 移动硬盘的USB接口如何选择**

！ 目前有些笔记本为用户提供了增强供电型USB接口，在这些USB接口上有“闪电”或“+”这样的图标，这些接口是厂家专为移动硬盘等需要提供较大电力支持的设备而准备的，因此，笔记本用户应尽量将移动硬盘移动硬盘中进行充电的USB设备连接到这个接口上使用。

在台式机方面，用户应该尽量使用主机后方的主板I/O面板上的USB接口。如果碰见了只能使用扩展USB接口的情况，则可以将Y型USB数据线的两头分别插在两个USB插针引出的USB接口上。因为现在很多主板都为每一组USB扩展插针提供了一个稳定电压的电容。当移动硬盘同时使用两组插针上的USB接口时，就有两颗这样的电容为移动硬盘的工作起移动硬盘保障作用。

## 基于DirectUI架构的中海油软件平台的用户体验设计与实现

中海油是中国海洋石油总公司的简称，是1982年成立的国家石油公司。经过几十年的不断发展，中海油拥有上百个油田工作系统，而每个油田系统均配有功能强大的软件系统作为它的支撑。每套软件系统的不断升级和维护让它们的程序变得更加复杂。由于这些软件都是从各种作业实践中逐步提炼而成的，从一开始就缺少统一的规划和全局的用户体验考虑，所以导致了一套系统一个样。给油田作业的工程师们带来了极大的操作难度。目前，中海油的软件系统的用户体验方面的问题已经严重影响到了海上作业的工作效率。中海油集团的上上下下也对用户体验问题高度关注。

UIPower在充分了解中海油的后台系统的需求后，建议中海油采用UI设计+UI开发的整体UI解决方案。

UI设计阶段：进行了用户研究、交互设计、用户体验测试和视觉设计等方面的工作。首先解决了软件难用的问题，对软件中的各种窗体进行了重新规划，详细分析了各种控件在实际使用中的频率。接着提出中海油作业软件的整体交互设计规范。从规范上确保未来的各个软件系统保持一致的操作习惯。通过对新的交互设计严格的实验室测试来检验它的科学合理性，去除里面不合理的地方，不断迭代最终形成易上手、易操作的软件交互模型。UIPower的视觉设计师根据多年的经验对软件进行整体风格、色调、质感等的创意，最终形成在视觉上具有高度专业性的软件高保真设计稿。

UI开发阶段：对于UI改造项目来说，设计出漂亮的界面其实已经很困难了，而要将专业的UI设计真正实现出来则是难上加难。原因是客户端各个开发工具对UI的实现均支持不足，无法满足各种新特效新质感的窗口或控件的交互和视觉呈现。一般来说，要实现高保真的效果图程序需要花费半年甚至更多的人月数。然而，现实却无法允许这样的界面开发进度。UIPower一直采用自主研发的DirectUI产品作为他们的界面开发的基石。DirectUI可以将一个软件的所有界面在1周内全部实现出来，而且对界面的效果做到没有损失，UIPower对项目验收的标准是“像素级比对”，即在最后的成型程序与高保真的视觉设计稿通过屏幕的逐像数比对，只要有一对像素的值不一样即认为是不通过验收。正是基于这样的验收标准，UIPower在业内迅速成为国内大型企业争相合作的对象。

经过2个多月的整体改造，中海油集团内的几款重要的软件系统，均已成功上线，并正式投入了使用。从集团上上下下的相关人员对本次的UI改造成果表示了高度的认可和极大的满意。

UIPower于2004年成立至今，一直以“让天下没有难做的界面“为使命，先后服务于华为、中兴、盛大网络、中国移动、中国电信、铁道研究院等国内大型企业。UIPower的阙总在接受采访时说：“未来UIPower一直致力于界面产品的研发与升级，全面助力中国软件相关企事业单位，提高中国软件产品在国际市场上的竞争力，实现软件强国的梦想！”。


书名：SAP MDM 主数据管理
ISBN：978-7-302-31787-6
定价：98 元
作者：和轶东、张怡、曹乃刚 著

本书结合了 SAP 官方最新技术以及作者在主数据管理领域持续多年的多个大型项目经验，通过从原理到架构、从功能到实现，深入浅出、图文并茂地介绍了 SAP MDM 主数据管理解决方案。

本书是 SAP 企业信息化与极佳实践丛书，该丛书是清华大学出版社与 SAP 合作的丛书，为权威的官方指南，伴随着大数据、云计算、移动化等先进技术的应用和推广，主数据管理在这个词在企业信息管理领域经常被谈起，且目前 SAP MDM 是一个较新的模块，国内对于熟练掌握该模块的顾问需求量日益增大，本书是介绍 SAP MDM 产品的一本中文书籍，内容完整、深度适当，既是一本叙述 MDM 功能的书，一本练习书，还可以作为配置速查手册。


书名：jQuery 高级编程
ISBN：978-7-302-31784-5
定价：48.00 元
作者：（美）劳伦斯（Larsen, R.）著

如果想创建出基于标准的交互型 Web 网站，就应该充分利用当前热门的新兴 Web 开发技术。在 Web 开发过程中将会遇到各种难题，本书着重介绍 jQuery 解决这些难题的“硬功夫”。突出介绍了 jQuery 核心库以及如何将 jQuery 集成到 Web 页面之中，此外还深入讲解 jQuery UI、jQuery 插件开发、jQuery 模板、JavaScript 设计模式等核心技术。经过本书的学习，读者将精通创建交互型网站关键任务所需的 jQuery 技术。

本书介绍了如何操作 DOM 元素和使用数据；使用 HTML 表单、AJAX 和 JSON 的步骤；介绍如何使用动画和设置 CSS 属性，为网站创建吸引眼球的效果；如何使用 jQuery 插件开发久经考验的极佳实践扩展 jQuery；编写高效 jQuery 代码、网站优化和扩展 JavaScript Object 的极佳实践。


书名：移动 Web 开发高级教程——使用 WordPress 、Joomla! 和 Drupal
ISBN：978-7-302-32078-4
定价：69.80 元
作者：（美）皮尔斯（Pearce, J.）著

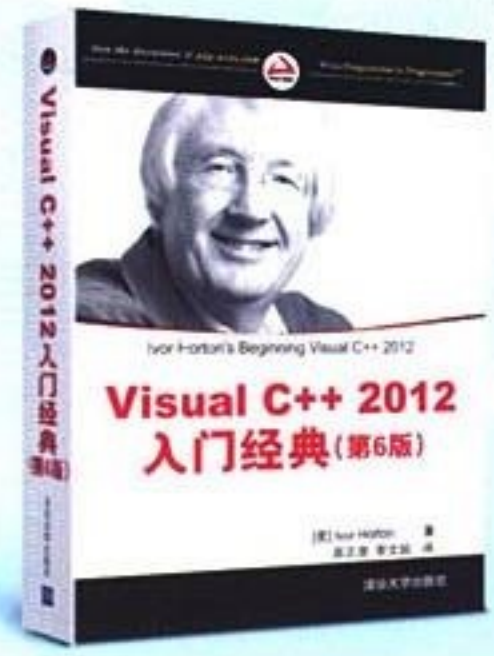
随着移动 Web 的日益流行，用户十分希望看到时尚华丽、精美流畅的网站。本书采用新颖独到的方式描述如何利用当今流行的 CMS 来开发移动内容并安装、配置、测试和集成移动网站，从而使移动用户获得愉悦舒畅的体验。本书浓墨重彩地描述 WordPress、Joomla! 和 Drupal 这三种特定平台，讲述如何结合每种 CMS 来设计移动网站，并分步介绍如何安装和配置插件、模块和主题，让您快速掌握基本建站技巧。

本书简要介绍了移动 Web 的发展历程和特点；分析移动 Web 技术要领；介绍移动设备、网络和二者面临的挑战；讨论移动 Web 的当前发展态势，以及如何合理地完成设计；介绍在执行 CMS 移动化之前应做的选择和决策；描述独立于平台测试、部署和集成移动网站的过程；讨论移动分析、网站宣传及其他运营主题。

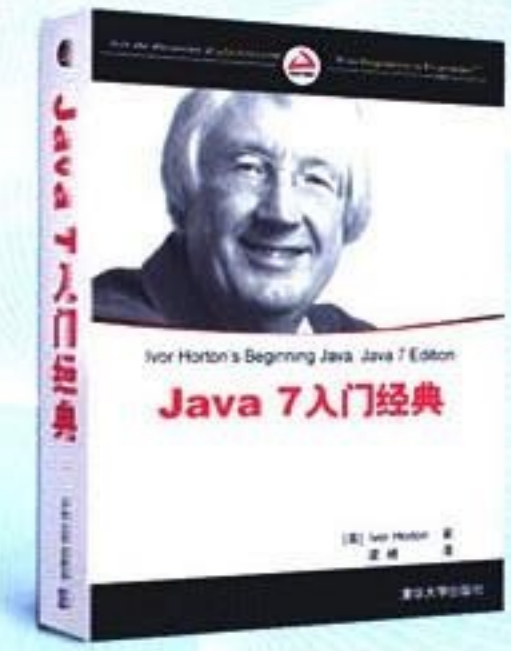

书名：iOS cocos2d 2 游戏开发实战（第 3 版）
ISBN：978-7-302-31892-7
定价：59.80 元
作者：（美）史蒂芬（Steffen Itterheim），（德）勒夫（Andreas Löw）著

本书是畅销书《iOS 5 cocos2d 游戏开发实战（第 2 版）》升级版图书，针对 cocos2d 2，源代码全面更新，本书详细介绍了 cocos2d 游戏引擎，关注的是创建完整 cocos2d 游戏的过程而不是展示大量的 iOS SDK 或 OpenGL 代码。在学习 cocos2d 2 游戏开发的过程中，你还会学到 cocos2d 游戏引擎中重要的编程概念并完成一些移动游戏开发的极佳实践，包括对精灵批处理技术、纹理地图、平行视差滚屏、触控和加速计输入。这有助你理解一些内部原理，当为交互式游戏进行设计、架构以及编写代码时，这些原理会帮你做出一些艰难决策，你能够从中获得许多宝贵的经验，进而在以后的项目中获得成功。

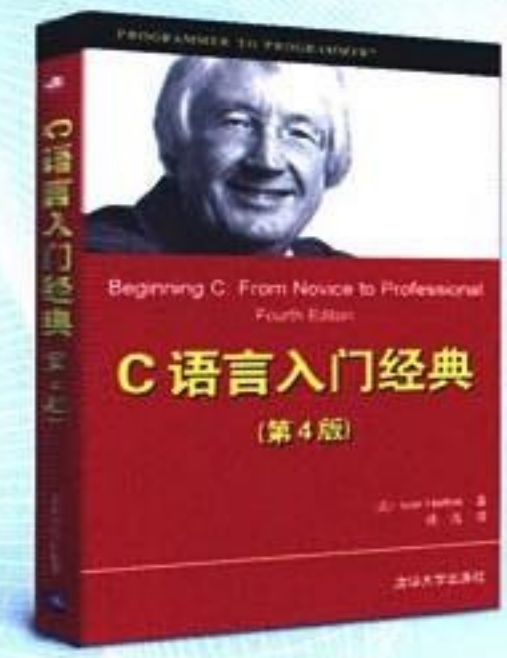
# 权威 编程经典 畅销书



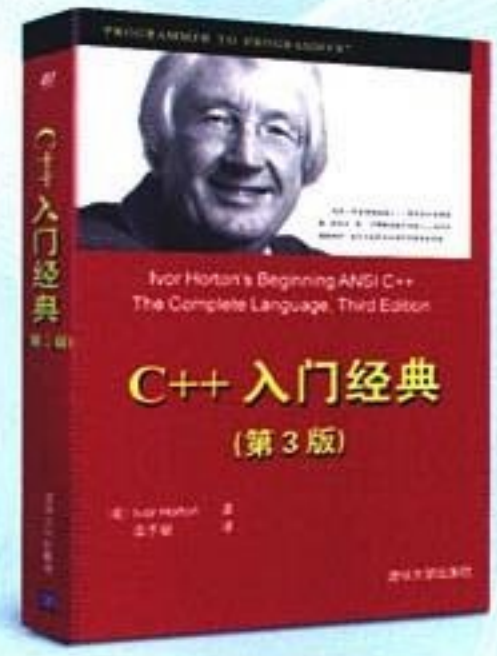

书号：9787302319009
定价：98.00元



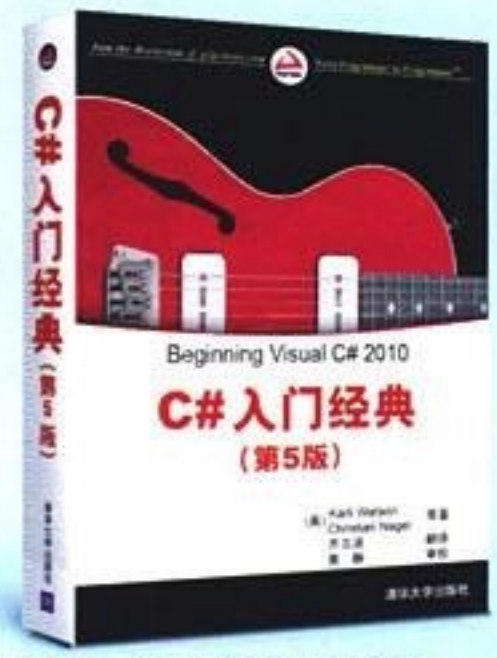

书号：9787302289593
定价：118.00元




书号：9787302170839
定价：69.80元



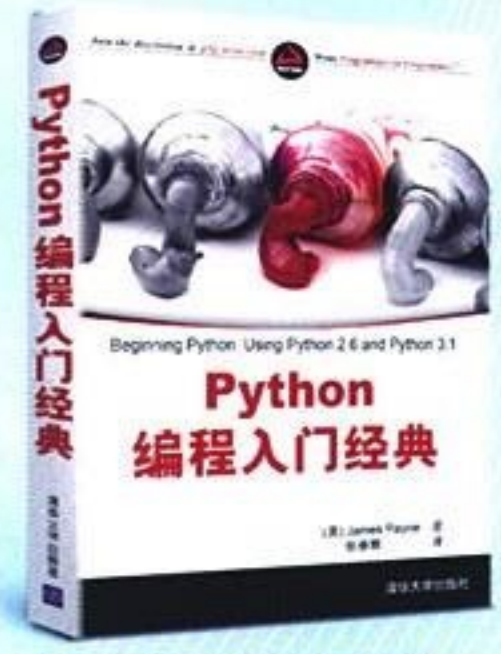

书号：9787302120629
定价：98.00元



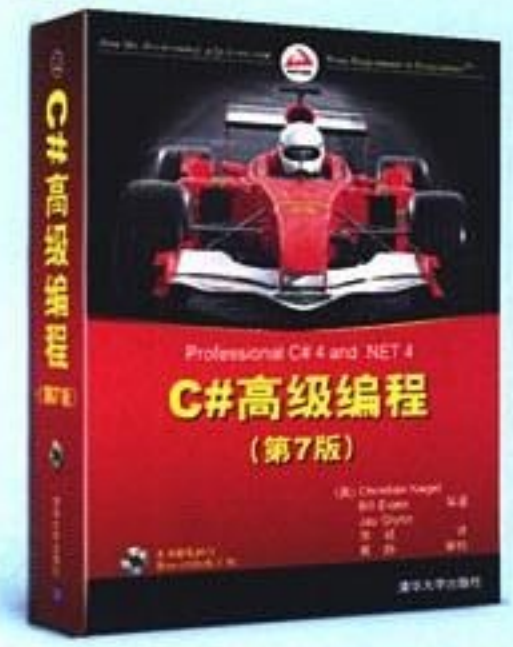

书号：9787302241300
定价：99.80元


书号：9787302245612
定价：88.00元

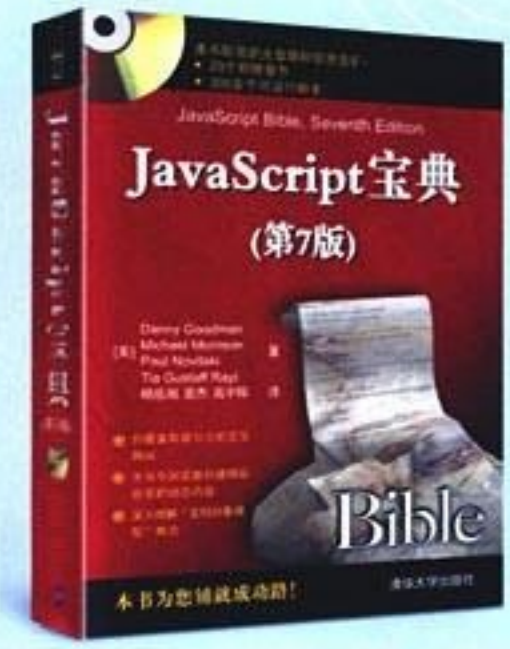

书号：9787302236962
定价：85.00元



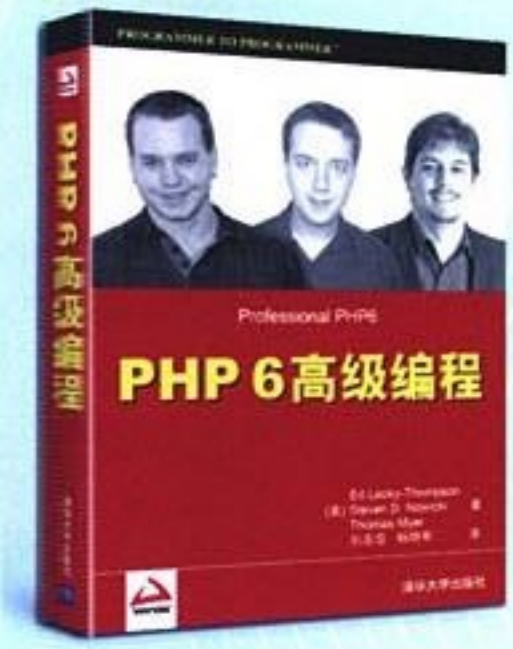

书号：9787302257097
定价：68.00元




书号：9787302239376
定价：148.00元

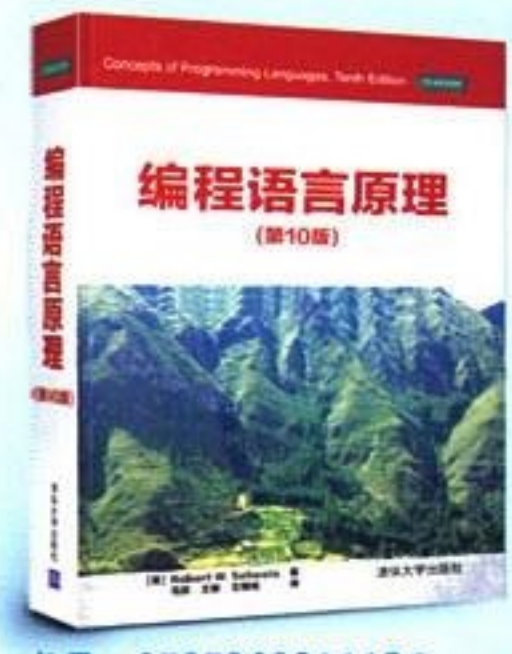



书号：9787302303220
定价：128.00元

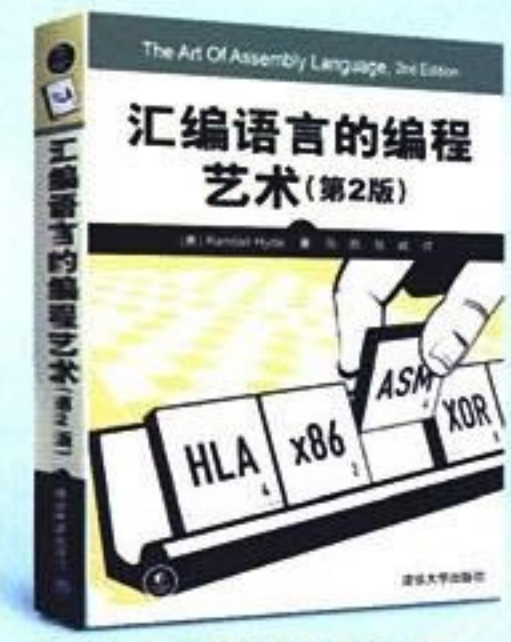



书号：9787302238249
定价：86.00元


书号：9787302273608
定价：98.00元




书号：9787302311126
定价：98.00元




书号：9787302263739
定价：69.80元


书号：9787302283669
定价：58.00元


书号：9787302282082
定价：59.90元

清华大学出版社
http://www.tup.com.cn

# 新一代 32位高性能智能卡加密锁

# Ucode3000

## 专注加密 匠心之作


一个开放的加密平台
灵活构建属于自己的加密体系
彻底杜绝各类加密锁的模拟、复制、逆向等破解行为

- 完备的国密、商密算法备调
- 灵活的自定义代码移植机制
- 强大的远程远程控制功能设计
- 安全的代码自毁机制

Ucode3000智能卡加密锁：

真正的32位智能卡核心加密锁，可以将关键代码从软件程序中“扣”出，移植进加密锁内，片上运行。硬件内部本身集成多套对称、非对称加密算法可供直接调用。Ucode3000是一个开放的加密平台，加密机制和加密硬件都提供了极高的加密强度，用户可自己规划和控制硬件内部程序流程，配合上层程序灵活构建各种各样的加密方案。强大的远程功能设计，支持功能密调、代码自毁，可远程精准、安全地控制和改变发出每只锁的内部加密机制，轻松实现加密锁激活、吊销、软件控制、加密方案调整、时间、次数的开放、禁止等等功能。

北京素志科技发展有限公司
电话：010-62275133 62275135
传真：010-62275135
网址：http://www.suciz.com
地址：北京市海淀区联慧路99号海云天D2005#